

AFRL-IF-RS-TR-2002-29
Final Technical Report
February 2002



PEER TO PEER INFORMATION SYSTEM MANAGEMENT

BBN Technologies

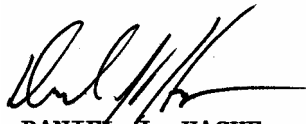
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK**

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-29 has been reviewed and is approved for publication.

APPROVED:



DANIEL J. HAGUE
Project Engineer



FOR THE DIRECTOR:

WARREN H. DEBANY, Jr., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE			<i>Form Approved</i> <i>OMB No. 074-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing this collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE Feb 02	3.. REPORT TYPE AND DATES COVERED Final Aug 96 – Sep 98	
4. TITLE AND SUBTITLE PEER TO PEER INFORMATION SYSTEM MANAGEMENT			5. FUNDING NUMBERS C - F30602-96-C-0049 PE - 62702F PR - 4519 TA - 22 WU - 40	
6. AUTHOR(S) Robert Coulter, Irvin Schick and Linsey O'Brien				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) BBN Technologies 10 Moulton Street Cambridge, MA 02138			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/IFGA 32 Brooks Road Rome, NY 13441-4505			10. SPONSORING / MONITORING AGENCY REPORT NUMBER AFRL-IF-RS-TR-2002-29	
11. SUPPLEMENTARY NOTES AFRL Project Engineer: Daniel J. Hague, IFGA, 315-330-1885				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.				12b. DISTRIBUTION CODE
13. ABSTRACT (Maximum 200 Words) This is a Systems Requirement Document for the Peer to Peer Information Systems Management to fulfill requirements of contract F30602-96-C-0049. It covers detailed restraints and requirements identified by the RFP SOW or identified to date by BBN as necessary to develop a system architecture that allows SOW-required management information transfer to be a securable, object-orientated infrastructure that treats the communicating parties as peers and optimizes according to information utilization and usage characteristics. A survey is included to provide the sponsor with a critical overview of the systems, standards, and technologies pertinent to the development of a generic peer-to-peer communications architecture for a network management system.				
14. SUBJECT TERMS Architecture Network Management System, Current Network Management Technologies, CORBA Integration				15. NUMBER OF PAGES 85
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL	
NSN 7540-01-280-5500			Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18 298-102	

Table of Contents

1.	EXECUTIVE OVERVIEW	1
1.1.	SCOPE.....	1
1.2.	DOCUMENT STYLISTIC CONVENTIONS	1
1.3.	INTRODUCTION AND OVERVIEW.....	1
1.4.	RELATED DOCUMENTS.....	2
1.4.1.	<i>Request for Proposal</i>	2
1.4.2.	<i>BBN Proposal</i>	2
1.4.3.	<i>Market and Technologies Study</i>	2
1.4.4.	<i>System Architecture</i>	2
1.4.5.	<i>Prototype Implementation Plan</i>	2
2.	DOCUMENT STATUS.....	2
3.	ENVIRONMENTAL REQUIREMENTS	2
4.	MANAGEMENT INFORMATION REQUIREMENTS.....	3
4.1.	MANAGEMENT APPLICATION OPERATIONS.....	3
4.1.1.	<i>Status Monitoring</i>	3
4.1.2.	<i>Event Monitoring and Management</i>	3
4.1.3.	<i>Historical Data and Trend Monitoring</i>	3
4.1.4.	<i>Control Commands</i>	4
4.2.	TARGET COMPONENTS.....	4
4.2.1.	<i>SNMP interfaces to IP Components</i>	4
4.2.2.	<i>XBind ATM management interface</i>	4
5.	SUPPORTING INFRASTRUCTURE REQUIREMENTS	5
5.1.	DISTRIBUTION	5
5.1.1.	<i>Peer to Peer Distribution</i>	5
5.1.2.	<i>WAN Topology</i>	5
5.1.3.	<i>Usage-based Distribution</i>	6
5.1.4.	<i>Standard Distribution Framework</i>	6
5.2.	COLLATERAL SERVICES	7
5.2.1.	<i>Object-based datatypes and access</i>	7
5.2.2.	<i>Integrated Object Directories</i>	7
5.2.3.	<i>Integrated Distributed Time Services</i>	7
5.2.4.	<i>Integrated Security</i>	7
5.2.5.	<i>Manageability of System Components</i>	8
5.2.6.	<i>Graphic User Interface</i>	8
5.2.7.	<i>Storage</i>	8
6.	EXECUTIVE APPENDICES.....	9
6.1.	REQUIREMENTS MATRIX	9
6.2.	ON-LINE REFERENCES	11
SECTION 1 - CURRENT NETWORK MANAGEMENT TECHNOLOGIES.....		13
7.	OVERVIEW	13
7.1.	NETWORK MANAGEMENT PLATFORM FOR PROTOTYPE PEER-TO-PEER SYSTEM.....	14
7.2.	CRITERIA	14
7.3.	CORBA INTEGRATION.....	15
7.4.	SUPPORT FOR JARGONS	15

7.5.	NOTES ON INITIAL DESIGN APPROACH.....	16
8.	INDUSTRY LEADERS AND PRODUCTS	17
8.1.	TIVOLI TME10.....	17
8.1.1.	<i>Environment</i>	17
8.1.2.	<i>Framework Services</i>	17
8.1.3.	<i>Event Services</i>	18
8.1.4.	<i>GUI</i>	18
8.1.5.	<i>Security</i>	18
8.1.6.	<i>Contacts</i>	18
8.1.7.	<i>Tivoli References</i>	18
8.2.	HP OPENVIEW	23
8.2.1.	<i>Environment</i>	23
8.2.2.	<i>Framework Services</i>	23
8.2.3.	<i>Event Services</i>	23
8.2.4.	<i>GUI</i>	23
8.2.5.	<i>Security</i>	23
8.2.6.	<i>Contacts</i>	24
8.2.7.	<i>HP References</i>	24
8.3.	BULL ISM/OPENMASTER.....	25
8.3.1.	<i>Framework Services</i>	25
8.3.2.	<i>Event Services</i>	25
8.3.3.	<i>Contacts</i>	25
8.3.4.	<i>Bull References</i>	25
8.4.	CABLETRON SPECTRUM	26
8.4.1.	<i>Environment</i>	27
8.4.2.	<i>Framework Services</i>	27
8.4.3.	<i>Interoperability with HP OpenView</i>	28
8.4.4.	<i>Event Services</i>	28
8.4.5.	<i>GUI</i>	28
8.4.6.	<i>Security</i>	28
8.4.7.	<i>Contacts</i>	28
8.5.	COMPUTER ASSOCIATES INTERNATIONAL INC. CA-UNICENTER	29
8.5.1.	<i>Environment</i>	29
8.5.2.	<i>Framework Services</i>	29
8.5.3.	<i>Event Services</i>	29
8.5.4.	<i>GUI</i>	29
8.5.5.	<i>Security</i>	29
8.5.6.	<i>Contacts</i>	29
8.5.7.	<i>CA References</i>	30
8.6.	SUN SOLSTICE ENTERPRISE MANAGER	31
8.6.1.	<i>Environment</i>	31
8.6.2.	<i>Framework Services</i>	31
8.6.3.	<i>Event Services</i>	31
8.6.4.	<i>GUI</i>	31
8.6.5.	<i>Contacts</i>	31
8.6.6.	<i>Sun References</i>	32
9.	INNOVATIVE COTS SYSTEMS.....	32
10.	STANDARDS ORGANIZATIONS.....	32
10.1.	DISTRIBUTED MANAGEMENT(DISMAN) GROUP OF THE IETF (INTERNET ENGINEERING TASK FORCE) 32	
10.2.	THE IEEE COMMUNICATIONS SOCIETY (IEEE COMSoc).....	32
10.3.	THE INTERNATIONAL FEDERATION FOR INFORMATION PROCESSING (IFIP)	32
10.4.	EWOS	32

10.5.NETWORK MANAGEMENT FORUM	33
11. VENDORS' USER GROUPS AND PARTNERSHIPS	35
11.1.HP OPENVIEW FORUM	35
11.2.TIVOLI 10/PLUS ASSOCIATION.....	35
12. CONFERENCES	35
12.1. IM '97 THE FIFTH IFIP/IEE INTERNATIONAL SYMPOSIUM ON INTEGRATED NETWORK MANAGEMENT 35	
12.2. DSOM, OCTOBER 28-30, 1996.....	37
13. RESEARCH PROJECTS	38
13.1.THE SIMPLE GROUP	38
13.2. ARPA ACTIVENETS PROJECT	39
13.2.1. Columbia University Management by Delegation Project	39
13.2.2. Columbia University NetScript Project.....	39
13.2.3. MIT Active Networks Project.....	39
13.2.4. University of Pennsylvania and Bell Communications Research SwitchWare Project.....	39
13.2.5. BBN ActiveNets Project	40
13.3. NEW ARPA INITIATIVE IN ACTIVE NETWORKS.....	40
13.4.NETWORK MANAGEMENT IN A MULTI-NATIONAL ENVIRONMENT	40
13.5.ACCORD PROJECT.....	40
14. TRENDS AND EMERGING TECHNOLOGIES	40
15. INDUSTRY COMMENTS	41
15.1.NETWORK COMPUTING ONLINE	41
15.2. COMMUNICATIONS WEEK	42
15.3.GARTNER GROUP.....	43
15.4. NETWORK MANAGEMENT MARKET	43
16. WEB SITES FOR MORE INFORMATION:.....	44
16.1.NETWORK MANAGEMENT	44
16.2.SNMP NEWS	44
16.3.TECHWEB.....	44
SECTION 2 - PEER TO PEER INFORMATION SYSTEM MANAGEMENT ARCHITECTURE. 45	
17. EXECUTIVE OVERVIEW.....	45
17.1. IDENTIFICATION	45
18. DOCUMENT STYLISTIC CONVENTIONS.....	45
18.1. DOCUMENT STATUS	45
18.2. SYSTEM OVERVIEW.....	45
18.3. DOCUMENT OVERVIEW	45
19. REFERENCED DOCUMENTS.....	46
19.1. CONTRACTOR SPECIFICATIONS	46
19.2. OTHER SPECIFICATIONS	46
20. SYSTEM-WIDE DESIGN DECISIONS	47
20.1. MANAGEMENT APPLICATION INTEGRATION.....	47
20.2. TARGET COMPONENTS INTEGRATION.....	47
20.3. OBJECT-ORIENTED ARCHITECTURE	48
20.4. DISTRIBUTION	48
20.4.1. Distribution Between Peers.....	48

20.4.2.	<i>WAN Topology</i>	48
20.4.3.	<i>Distribution of Management Control and Information</i>	49
20.4.4.	<i>Standard Distribution Framework</i>	49
20.4.5.	<i>Coping with CORBA deficiencies</i>	49
20.5.	SECURITY INTEGRATION	51
20.6.	SYSTEM MANAGEMENT	51
20.7.	STORAGE	51
20.8.	GRAPHIC USER INTERFACE	52
21.	FUNCTIONAL COMPONENT DESCRIPTION	52
21.1.	BASIC MANAGEMENT OF SHARED OBJECTS	54
21.2.	ASYNCHRONOUS ATTRIBUTE MONITORING	55
21.3.	SHARED HISTORICAL CONTEXT DATA	55
21.4.	SHARED OBJECT CONTROL OPERATIONS COORDINATION	55
21.5.	COLLECTIVE EVENT SOURCES	56
21.6.	COMPETING MULTIPLE SOURCES	56
22.	SYSTEM INTERFACE DESCRIPTION	57
22.1.	JARGON-BASED DISTRIBUTION	57
22.1.1.	<i>Direct IDL Jargons</i>	59
22.1.2.	<i>PASS-based Jargons</i>	60
22.1.3.	<i>Jargon Management and Shared Managed Objects</i>	66
22.2.	ASYNCHRONOUS ATTRIBUTE MONITORING INTERFACE	68
22.2.1.	<i>Status_PASS API</i>	69
22.2.2.	<i>Status_PASSREADER</i>	69
22.2.3.	<i>Status_PASSWRITER</i>	69
22.2.4.	<i>STATUS_PASS IDLs</i>	69
22.3.	HISTORICAL DATA COLLECTION AND DISTRIBUTION	69
22.3.1.	<i>TrendTable by Time</i>	71
22.3.2.	<i>TrendTable by Device</i>	71
22.3.3.	<i>TrendTable by Variable</i>	71
22.3.4.	<i>Common Trend Datatypes IDL</i>	72
22.3.5.	<i>Trend Bulk IDL</i>	72
22.3.6.	<i>Trend Bulk Factory IDL</i>	75
22.3.7.	<i>Trend Subscription API</i>	76
22.4.	ATTRIBUTE SNAPSHOTS	76
22.5.	CONTROL COMMANDS	77
22.5.1.	<i>Control IDL</i>	77
22.6.	EVENT MONITORING	78
22.6.1.	<i>Event_PASSREADER</i>	78
22.6.2.	<i>Event_PASSWRITER</i>	78
22.6.3.	<i>EVENT_PASS IDL</i>	78
22.7.	COLLATERAL SERVICE INTERFACES	79
22.7.1.	<i>Directory Services</i>	79
22.7.2.	<i>Time Service</i>	79
22.7.3.	<i>Security Services</i>	79
22.7.4.	<i>Management Services</i>	79
22.7.5.	<i>Persistent Storage Services</i>	80
22.8.	USER INTERFACES	80
23.	SYSTEM COMPONENT DESCRIPTION	80
23.1.	STATUS TRANSFER	82
23.1.1.	<i>STATUS_PASS</i>	82
23.1.2.	<i>STATUS_PASS Writer</i>	82
23.1.3.	<i>STATUS_PASS Reader</i>	82
23.1.4.	<i>TME Unwrapper</i>	82

23.1.5.	<i>HPOV Wrapper</i>	82
23.1.6.	<i>HPOV Unwrapper</i>	82
23.1.7.	<i>TME Wrapper</i>	83
23.2.	TREND / HISTORICAL CONTEXT BULK TRANSFER	83
23.2.1.	<i>TME Unwrapper</i>	83
23.2.2.	<i>HPOV Wrapper</i>	83
23.2.3.	<i>HPOV Unwrapper</i>	83
23.2.4.	<i>TME Wrapper</i>	83
23.3.	TREND / HISTORICAL CONTEXT SUBSCRIPTION TRANSFER	84
23.3.1.	<i>TREND_PASS</i>	84
23.3.2.	<i>TREND_PASS Writer</i>	84
23.3.3.	<i>TREND_PASS Reader</i>	84
23.3.4.	<i>TME Unwrapper</i>	84
23.3.5.	<i>HPOV Wrapper</i>	84
23.3.6.	<i>HPOV Unwrapper</i>	85
23.3.7.	<i>TME Wrapper</i>	85
23.4.	BASIC SHARED MANAGED OBJECT ATTRIBUTE SNAPSHOT	85
23.4.1.	<i>TME Unwrapper</i>	85
23.4.2.	<i>HPOV Wrapper</i>	85
23.4.3.	<i>HPOV Unwrapper</i>	85
23.4.4.	<i>TME Wrapper</i>	85
23.5.	BASIC SHARED MANAGED OBJECT ATTRIBUTE CONTROL	85
23.5.1.	<i>TME Unwrapper</i>	86
23.5.2.	<i>HPOV Wrapper</i>	86
23.5.3.	<i>HPOV Unwrapper</i>	86
23.5.4.	<i>TME Wrapper</i>	86
23.6.	EVENT FORWARDING	86
23.6.1.	<i>EVENT_PASS</i>	86
23.6.2.	<i>EVENT_PASS Writer</i>	86
23.6.3.	<i>EVENT_PASS Reader</i>	86
23.6.4.	<i>TME Unwrapper</i>	87
23.6.5.	<i>HPOV Wrapper</i>	87
23.6.6.	<i>HPOV Unwrapper</i>	87
23.6.7.	<i>TME Wrapper</i>	87
23.7.	JARGON MANAGEMENT.....	87
24.	APPENDICES.....	89
24.1.	ON-LINE REFERENCES	89
24.2.	GLOSSARY.....	89
24.2.1.	<i>Peer to Peer</i>	89
24.2.2.	<i>Jargon</i>	89
24.2.3.	<i>Managed Object</i>	89
24.2.4.	<i>Managed Object Instance or MOI</i>	89
24.2.5.	<i>Shared Managed Object</i>	90
24.2.6.	<i>Infrastructure Object</i>	90
24.2.7.	<i>Direct Object</i>	90
24.2.8.	<i>Indirect Object</i>	90
24.2.9.	<i>PASS Object</i>	90
24.2.10.	<i>Collateral Services</i>	90
24.2.11.	<i>Authentication</i>	90
24.2.12.	<i>Authorization</i>	90
24.2.13.	<i>Integrity</i>	90
24.2.14.	<i>Privacy</i>	91
24.2.15.	<i>Replay Attack</i>	91
24.2.16.	<i>Denial of Service Attack</i>	91
24.2.17.	<i>Persistent Storage</i>	91

24.2.18.	<i>Management of Management</i>	91
24.2.19.	<i>Heartbeat</i>	91
24.2.20.	<i>XBind</i>	91

Table of Figures

Figure 21-1	A Management System	53
Figure 21-2	Managed Object Structure and Formal MIB Definition	53
Figure 21-3	Shared Managed Objects.....	54
Figure 22-1	Jargon-based Shared Managed Objects.....	58
Figure 22-2	Example PASS Jargon Usage	65

1. Executive Overview

1.1. Scope

This document is submitted in fulfillment of requirement 4.1.2, and CDRL A005; from the Statement of Work for the Peer-to-Peer Information System Management Contract #F30602-96C-0049.

1.2. Document Stylistic Conventions

This document is published both in paper and hypertext formats. Double underlines represent Hypertext links in the paper form. The cross-references are identified inline, external references are collected in an Appendix, *On-line References*.

1.3. Introduction and Overview

This is the *System Requirements Document* for the Peer to Peer Information System Management program. It covers in detail the constraints and requirements identified by the RFP SOW or identified to date by BBN as necessary to develop a system architecture that allows SOW-required management information transfer:

1. to be within a securable, object-oriented infrastructure that treats the communicating parties as peers,
2. to be optimized according to the information's utilization and usage characteristics,

The minimum subset of management information to be transferred comprises

1. status monitoring
2. events, traps and notifications monitoring and management
3. control commands
4. monitoring within an historical context

The minimum subset of managed object classes to be covered comprises

1. IP devices
2. ATM devices.

Some of the capabilities will be supplied by purchased components; other capabilities will be provided by BBN-developed components. The specification of the components and their responsibilities is provided in the *Peer to Peer System Architecture document*.

1.4. Related Documents

1.4.1. Request for Proposal

Solicitation # F30602-96-R-0049

1.4.2. BBN Proposal

P96-STD-395

1.4.3. Market and Technologies Study

BBN Technical Report 8180, Nov. 1996

A report of the results of a survey covering the existing commercial off the shelf products and available technologies.

1.4.4. System Architecture

A specification of the system architecture based on the *Market and Technologies Study* results and this *Requirements Document*.

1.4.5. Prototype Implementation Plan

A specification of those aspects and components of the system architecture that shall be included in the proof-of-concepts prototype system.

2. Document Status

Delivered to Customer.

Last updated on 05/15/97 12:34:57 PM by Linsey O'Brien.

3. Environmental Requirements

The system shall operate on or in conjunction with at least the following platforms:

- Solaris 2.5.1
- HP Openview
- Tivoli TME

HP Openview was designated a required network management platform by the RFP Statement of Work sections 2.1 and 4.1.4. It is a legacy object oriented, remote network management system. It is not CORBA compliant.

Tivoli TME was selected to be the second of two network management platforms during the Market and Technologies Study. It was picked because it appeared to offer a solid, open technical foundation that also had significant COTS market share. It is CORBA 1.0 and MIB-II compliant, but the CORBA framework is not directly accessible except under stiff licensing fees. Technical strategies for dealing with these issues are described in the System Architecture.

Per conversations with the client, Rome Labs, The Peer to Peer Systems Architecture **shall not** be CMIS- or CMIP- based, although integration with CMIS components shall not be prevented.

4. Management Information Requirements

The following requirements are the management information factors driving the design of the Peer to Peer Network Management System Architecture.

4.1. Management Application Operations

The system shall support integrating significant management information input and output necessary to coordinate peer management operations. Such management information types included status, event, and statistical monitoring and management, and issuing control commands.

4.1.1. Status Monitoring

Status information is the heart of day to day operations. Monitoring status of critical components is the starting point for most management procedures. Integrating two peer systems' status information is primarily a matter of distributing updates among peers when the status of monitored components changes. This may be done through several distribution mechanisms: status polls, event management and monitoring the results of control operations. Which mechanism is used depends on how the network management systems in question classify the status information. Integrating those mechanisms is discussed in the Usage-based Distribution section below.

4.1.2. Event Monitoring and Management

Event information is often simply status change information and as such is distributed through an asynchronous notification mechanism. It is included here as a distribution variant of status monitoring. However, it can also convey other information with similar distribution characteristics.

4.1.3. Historical Data and Trend Monitoring

Historical data are generally accessed for one of two reasons:

1. Monitoring in Context: Data are collected and analyzed in real time in order to make network management decisions regarding routing, whether or not to dial-up additional bandwidth, when to run particular applications, and so forth. Although this function emphasizes current performance, it is generally not sufficient to merely query and receive a single number---such as the current utilization of a

particular link---since such data are meaningless unless placed within the broader context of trends and past performance.

2. Performance analysis: Data are collected and analyzed off-line in order to assess the health of the network, do capacity planning, bill users, and so forth. This function stresses historical data from which trends and use patterns must be extracted. The quantities of data may be very large, but their transfer is not very time-sensitive since performance analysis is usually conducted in batch mode during off-peak hours.

In either case, there are both implied distribution and storage requirements, covered in the Usage-based Distribution and Storage sections below.

4.1.4. Control Commands

Control commands are what complete the management loop and while they are the most powerful operations in the management system array, their fundamental distribution and storage requirements are trivial subsets of the monitoring operations. They do have, however, significant manageability and security requirements described in the relevant sections below.

4.2. Target Components

Monitoring and control operations require target components and those components further refine the type and usage of the operations and their management information.

4.2.1. SNMP interfaces to IP Components

The system shall allow information and functions exported by SNMP V2 network management systems to be incorporated.

The system shall support the networking MIBs defined according to the IETF SNMP V2 standards to the extent that the peer network management system being integrated supports such MIBs. Other MIBs, if defined according to STD 17 / RFC 1213 or RFC 1902, may be integrated if the management information system allows MIBs and managed object classes to be added.

4.2.2. XBind ATM management interface

The system shall allow information and functions exported by XBind network management systems being developed at Columbia University to be incorporated.

5. Supporting Infrastructure Requirements

The following factors are requirements derived from the need to integrate with other existing or planned technologies.

5.1. Distribution

The primary issue in integrating peer management systems is defining how their information distribution systems will be connected such that information from one system can easily and appropriately flow to one or more others. The following requirements affect connecting two or more management systems' distribution infrastructure.

5.1.1. Peer to Peer Distribution

The system shall integrate management information and functions from a variety of interfaces such that the operation of one management system shall not be required in order for the other to operate. However, if the operators of each system so allow, one system can monitor, control, exchange historical data with, and field events from the other system. The first peer system integrated shall be based on HP Openview. The second peer system integrated was selected as a part of the study phase and is the Tivoli TME system.

5.1.2. WAN Topology

As networks expand, they are increasingly having to interoperate over Wide Area Network (WAN) boundaries with peer networks. Such peer networks are run by peer organizations using their own network management systems. Peer management systems are often not located within the same [extended] LAN because peer business organizations managed by peer but different management systems generally do not share a LAN. Support for management information distribution across such borders and WAN gateways is common and therefore required.

One common exception to this rule of peer-management situations used to be the rule. It occurs when a single organization is trying to migrate gradually from one management system to another. Often the two management systems are peers during the transition. This was the most common peer management situation when most networks were extended LANs. Even if they had WAN components they were still managed by a single business organization. Given that the presumption that one system was replacing the other, the systems were only temporarily peers, and market share was changing hands, most network management vendors opposed peer management configurations that would make it easy for another vendor to replace them and their customers generally condoned the resulting technical, architectural and financial barriers to such configurations.

Such situations will continue to exist; large company / large market share vendor opposition to peer management architectures may be defused by allowing peer communications *only* across WANs. This will restrict the potential takeover targets to larger organizational groups and minimize the risk that a competitor will gain a toehold.

As smaller companies will want to gain such a toehold, they too should support the WAN distribution requirement.

A third driving force for WAN distribution is the nature of ATM equipment management interfaces (another Peer to Peer system requirement, see the section on XBind / ATM components below). In many cases the ATM switches do not provide direct interfaces to the relevant operations and data and network management systems (such as XBind) export the management data. When such systems setup ATM connections they may have to do so across a WAN channel because that is often the initial link between organizations deploying ATM equipment, much like a lightweight heaving line is used to pull a hawser across or electrical fish tape is used to pull cable.

5.1.3. Usage-based Distribution

The system shall provide various mechanisms in order to distribute management information and function based on the datatype and access usage of the management information concerned. The usage types shall include representatives of each of the RFP SOW categories plus any others judged necessary to demonstrate the utility of the Peer to Peer System Architecture.

The first usage type supported shall be for monitoring and control that can be bandwidth-intensive, either because the component information is volatile and a synchronous API would be inefficient or because the response is potentially very large. The second usage supported shall be for non-bandwidth intensive monitoring and control commands. The third type supported shall be for large bandwidth intensive data sets. The fourth usage supported shall be for events, alerts, traps or other ephemeral and asynchronous notifications.

For example, the two historical monitoring functions described earlier impose distinct distribution requirements. Monitoring in Context requires a synchronous mode in which a current measurement, as well as some statistics (e.g. very recent history, median, 5th and 95th percentiles) that allow it to be placed within the appropriate context, are queried and received. Performance Analysis requires an asynchronous mode in which users subscribe to particular historical measurement data sets that are potentially quite large, and receive them periodically (e.g. daily).

5.1.4. Standard Distribution Framework

While proposing a collection of usage-optimized communication 'jargons' to distribute management control and data among peer systems, we do not want to create a set of single-use spaghetti strands drooping between systems. Therefore, we consider a distribution standard necessary to minimize the number and methods of low-level mechanisms used for distribution. These low-level standard capabilities will then be utilized in a variety of standardized ways to support usage-optimized communication between the management systems. Such a standard must also support the object-oriented requirement covered in section 4.2.1 .

5.2. Collateral Services

Hooking together information system distribution infrastructures often generates requirements based on the integration process and also requires additional services not directly or immediately concerned with management information transfer operations although they often make it possible. Such collateral services requirements are addressed in this section and include object-oriented development environments, distributed directory (naming) services, distributed time services, security and access control services with network-wide coverage, managed object management and configuration services, etc.

5.2.1. Object-based datatypes and access

The system shall support object-based datatypes and provide object methods so as to support access to management information from a wide variety of network components and management applications without having to explicitly specify each individual interface.

5.2.2. Integrated Object Directories

The system shall provide an integrated object directory for all managed objects shared among peer systems such that it can be managed from each system using a standard identifier. Such a directory may be manually populated and its naming conventions based on local agreements.

5.2.3. Integrated Distributed Time Services

The system shall provide an integrated time service for management operations, historical information and logs such that timestamped information from one source may be compared and sorted with timestamped information from another peer system. Such integration may be provided by a manually generated mapping between local time zones and local agreements about clock usage and time providers.

5.2.4. Integrated Security

The system shall define and provide hooks for security mechanisms provided by Odyssey Research Associates at all necessary levels. In particular it shall provide hooks that allow management systems to use modular security mechanisms that:

- **authenticate the source and preferably the destination** of status updates, control commands, events, and historical data.
- **authorize the client application** which issues control commands or which attempts to supply input to a management application
- **ensure the integrity** of such management traffic
- **preserve management information integrity**

Authentication, Authorization and Data Integrity mechanisms shall be supplied by the designated security partner, Odyssey Research Associates.

Derived requirements arising from the need to extend integrated security to WAN environments via common security mechanisms such as firewalls will be considered Management of Security of Management component issues and again, hooks only will be defined and provided.

5.2.5. Manageability of System Components

The system itself shall have a management interface such that an authorized person can review the management system configuration information and make necessary changes. The management interface shall integrate component management system management interfaces and BBN supplied integration components, using, as much as possible, the same or similar GUI environments. Installation and other similar situations which are prone to bootstrapping dependency issues may use methods other than an integrated GUI to minimize the user effort and learning curve.

5.2.6. Graphic User Interface

The system GUI shall be usable on the same screen at the same time as the GUI supplied by the network management platform (HP Openview or Tivoli TME). Where appropriate, the system GUI should integrate with that of the network management platform so that, for example, system functions can be invoked via additional menu items or menus that appear as part of the network management platform's GUI

5.2.7. Storage

In addition to distributing management information, each type of usage has distinct storage requirements. For example, the two historical monitoring functions, Monitoring in Context and Performance Analysis described earlier and the Event Logs implied by event monitoring also have storage requirements.

1. In the case of monitoring current measurement information in context, the information becomes obsolete fairly rapidly. Thus, only the most recently queried information must be cached, and only for a limited period. To integrate two peer systems means aligning cache purging procedures or providing a buffering mechanism.
2. In the case of historical performance data or event logs, potentially large amounts of data must be stored for potentially extended periods of time. This may be done in a centralized or a distributed database. Transparency to the end-user is an important requirement. To integrate two peer systems means providing consistency and update mechanisms.

6. Executive Appendices

6.1. Requirements Matrix

The following matrix matches requirements listed in the Rome Lab's RFP Statement of Work.

Section	Type	Requirement
1.1	Arch	Peer to Peer NM Entity Comms
	Arch	Monitoring, including Status
	Arch	Control
	Arch	Fault Isolation
	Arch	Multiple NM Entities
	Arch	Info Flow Optimization
	Design	Rome Testbed integration Demo
2.1 (also 4.1.1, 4.1.2)	Study & Design	P2P communications
(also 4.1.1)	S&D	Applicable Standards
(also 4.1.1)	S&D	Applicable Existing Technologies
(also section 4.1.4)	Design	HP Openview
(also section 4.1.4)	Design	non-HP NM platform
3.1	Arch	End to End Resource Mgmt.
	Arch	Integrated Security
	Arch	ATM as Network Resources
4.1		
4.1.1	Sched	Deliver Study Results
4.1.2	Sched	Deliver Requirements Doc Req Delivery Plan
4.1.3	Sched	Deliver Arch Doc Doc Delivery Plan
4.1.3.1	Arch	Definitions of NM Entities: Functionality &

		Responsibilities (e.g. collection, storage, analysis/inference, display from 3.1
4.1.3.2 & 4.1.3.4	Arch	Definition of NM Entity Inter-relationships & Mechanisms (e.g. equal peers, hierarchical and the Mgmt. Info formats)
4.1.3.3	Arch	Definition of System I/F to Peer Mgmt. System
4.1.4		Deliver Implementation Plan (Design Doc.)
	Design	Exchange Mgmt Monitoring & Control Info.
	Design	HP Openview Usage
	Design	non-HP NM platform Usage
4.1.5	Sched	Implement Design sections)
4.1.5.1	Sched	Deliver minimum HP and non- HP platform configurations to Rome 1 month after approval
4.1.5.2	Sched	Deliver Phased Releases of Impl Starting with 4.1.5.1
4.1.6	Sched	Two Proof of Concept Demos
4.1.6.1	Sched & Impl	Demo Exchange of Monitoring Info, including Status
4.1.6.2	Sched & Impl	Demo Exchange of Monitoring Info, including Status and Control Info
	Sched	Deliver Source, Executables, Libraries and Tools (all properly licensed) using 3 or fewer BBN personnel in 5 days or less.
4.1.7	Sched	Reports

4.1.7.1	Sched	Monthly Status Reports
4.1.7.2	Sched	Funding Status Reports as required
4.1.7.3	Sched	Research Results and Evaluations as Scientific & Technical Reports, and Tech. Info Reports, as required plus a final S & T
4.1.7.4	Sched	Technical Status Presentations
4.1.7.5	Sched	Attend Technical Interchange Meetings

6.2. On-line References

The following are the on-line references elsewhere in this document; they will become stale over time and should be treated as such by January, 1998. In some cases, locally-stored snapshots have been made to minimize loss of the reference. Paper copies of snapshots of non-project documents will be provided at the end of the project .

P2P Study/Survey

<http://morpheus.bbn.com/p2p/rcsDocs/nm.html> .

P2P System Architecture

<http://morpheus.bbn.com/p2p/rcsDocs/sysarch.html> .

XBind

<http://www.ctr.columbia.edu/comet/xbind/xbind.html> .

SNMP V2 [IETF]

<http://www.ietf.org/ids.by.wg/snmpv2.html> .

SNMP V2 [BBN]

<http://morpheus.bbn.com/p2p/rcsDocs/snmpv2.html> .

IETF MIBs [IETF]

- RFC 1902 which defines the SMI, the mechanisms used for describing and naming objects for the purpose of management.
- STD 17, RFC 1213 defines MIB-II, the core set of managed objects for the Internet suite of protocols.

- SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, January 1996.
- M.T. Rose (editor), Management Information Base for Network Management of TCP/IP-based internets, Internet Working Group Request for Comments 1213. Network Information Center, SRI International, Menlo Park, California, (March, 1991).

Odyssey Research Associate
<http://www.oracorp.com/> .

Section 1 - Current Network Management Technologies

7. Overview

This survey of network management tools is submitted in fulfillment of requirement 4.1.1 and CDRL A004 from the Statement of Work for the Peer-to-Peer Information System Management Contract #F30602-96C-0049.

There are two primary goals for this survey:

First, to provide the sponsor with a critical overview of the systems, standards, and technologies pertinent to the development of a generic peer-to-peer communication architecture for network management systems (NMS).

Second, to select an appropriate network management tool to use in the prototype peer-to-peer network management system to be developed in the next phase of the project. In the Statement of Work, the sponsor identified HP OpenView as one of two systems to use in the prototype; the choice of the other system is to be recommended by BBN, as a result of this survey.

This survey was conducted in order to meet several objectives for the critical overview of the current state of network management systems:

1. To determine what software products and technology are widely available including determining which products are leaders in the market place.
2. To determine which companies and products are providing leading edge or innovative solutions.
3. To determine what organizations are actively involved in developing standards.
4. To identify projects at universities or other organizations involving leading edge research in the area of network management.

The survey addresses each of these areas and concludes with a recommendation for the network management system to be used in the prototype peer-to-peer communications system to be designed and implemented in the next phase of this project.

As network management technology and tools are rapidly evolving, this survey has been delivered in two forms: the standard, hard copy report and a document on the world wide web. In the latter form, the survey includes references to relevant web sites from industry, education, and research and standards organizations. In that form, the survey is a "living" document, providing up-to-date information more current than the publication date of the hard copy document. As the location and availability of web documents may change over time, all referenced web pages have been copied to a server at BBN where they will be maintained for the life of the contract. Each of these web pages contains the location of the original web page.

7.1. Network Management Platform for Prototype Peer-to-Peer System

We recommend Tivoli as the Network Management Platform to integrate with HP OpenView in the Prototype Peer-to-Peer System to be developed in the next phase of this project.

7.2. Criteria

The criteria for selection of a second network management system for the prototype phase of the project are both technical and non-technical.

The non-technical criteria include:

- Availability -- Can the system be obtained by other organizations that might be interested in making use of, or pursuing further research on, our work?
- Support -- Is the system currently supported? Is the vendor committed to support and improve the product for the foreseeable future?
- Manageability -- Is the vendor available for technical support? (On this point, vendors were judged by how well they responded to requests for information for this survey.)
- Environment -- Can the system be run on a Sun or HP workstation?
- Performance -- Will the system perform reasonably well, not only in a prototype environment, but also in a real-world network?

We first applied the above non-technical criteria, primarily through literature searches and searches of online information, to choose a small number of systems from the many network management systems available.

In applying the non-technical criteria, we decided to restrict further consideration to the leading COTS vendors. These vendors meet the criteria of availability and support. Further, these vendors have demonstrated that their products can perform in real-world environments. All the vendors considered deal with performance limitations by subdividing the overloaded domain and installing additional managers. Managing such a configuration may be difficult in very large networks, but this strategy is very common. Finally, and perhaps most importantly, integration of a widely available COTS product in a peer-to-peer management scheme will validate this methodology in a way that freeware or customized software could not.

After narrowing our choices to a few leading COTS vendors, we arranged for presentations by vendors and studied the product technical manuals.

The primary technical criteria for selection of a network management platform for the prototype phase of the project are:

- Ease of, or support for, CORBA integration, including:
 - CORBA wrappers for non-CORBA, proprietary object APIs
 - ORB interoperability for CORBA compliant object APIs
 - IDL back-end support for CORBA compliant object APIs for C++
- Support for the 'jargons' (events, status, control, trend) outlined in the proposal

7.3. CORBA Integration

One of the purposes of this project is to demonstrate how the old "request/response" approach to distributing management information is insufficient to cover the full range of management activities. We will use CORBA to create "request/response" objects (for control information), composite "stream" objects (for status and other monitoring), and other jargon objects that are deemed necessary. These different types of objects will be able to coexist in a CORBA framework, as will be demonstrated in the prototype phase of the project.

Any secondary platform, in addition to the HP OpenView platform required by the sponsor, should provide a CORBA V2.0 compliant developer's environment for on-site or third-party applications. CORBA will be used as the basis for the manager-to-manager integration. Although a CORBA interface is not strictly necessary, additional work would be required to "CORBAsize" objects to be imported/exported and integrated across platforms. As HP OpenView does not offer a CORBA API, the interface to this platform will include CORBA wrappers for shared objects.

Assuming a CORBA requirement, the leading COTS products are ranked as follows, based on their application interfaces:

- Tivoli TME10 (native CORBA)
- Cabletron Spectrum (proprietary, open, object-oriented APIs, with promised future support for CORBA)
- Computer Associates Unicenter (proprietary, object-oriented APIs, but no CORBA support)
- Sun Solstice (CMIS API; plans to support Java API, but no immediate plans to support CORBA)
- Bull OpenMaster (CMIP APIs)

7.4. Support for Jargons

We studied the Application Programming Interfaces of the various tools in an attempt to decide how well they would support the jargon approach. Questions such as "Does the NMS provide Event processing?" "How does the NMS allow application developers to define objects so that we can translate GETs and SETs?" "Do we have to map GET-RESPs into Events on the second system because the second system didn't issue a GET-REQ?" became selection criteria when we attempted to see how well our initial system architecture would be supported. In this preliminary architecture, managers will exchange a subset of the jargons as follows:

Events will be exchanged by mapping an EVENT message from NMS 1's managed object through a CORBA Event Stream into an EVENT message in NMS 2.

Status will be exchanged by mapping a sequence of GET-RESPs, which are replies to NMS 1's polls, into a MIST stream object, and then mapping these into a sequence of GET-RESPs to NMS 2's events/alarms/alerts.

Control will be supported by forwarding, via a proxy object, the SET-REQ from NMS 1 to NMS 2, which will then send it as the "real" object.

The exchange of trend data will be supported by forwarding a GetIFFLog from NMS 1 through a CORBA transport object to NMS2, then forwarding the RespIFFLog back through a (potentially different) CORBA transport object.

CORBA Transport objects are different than CORBA Management objects in that a Management object's IDL defines a MIB-based Request/Response interface, and a Transport object's IDL typically defines stream or other transport-based characteristics. Transport objects distribute a particular type of management information (the MIB's event in the case of the Event Stream, the MIB's status attribute in the case of the MIST, the IFF file in the case of the Trend transporter). Transport objects will enable the prototype system to work around the Request/Response communication paradigm that CORBA normally uses, while staying within the CORBA framework for integration purposes.

In summary, the network management platform chosen must allow

- the definition of MIB-based gateway objects in the network management platform and matching CORBA components, and
- the definition of transport-based application objects that convey Events, Status, Control and Trend information, between the source NMS MIB object (or application in the case of the Trend Logs), and the destination application. Destination applications include Logger for Events, Monitor/Display for Status and other monitoring items, and Trend Analysis/Logger for Trend Logs.

Several platforms support definition of MIB-based objects in their native mode. Only one platform, Tivoli, is CORBA-native. Definition of a CORBA half-gateway for non-CORBA-native platforms is platform-specific. This support will have to be provided for HP OpenView, which is not CORBA-native. The OpenView integration will prove the concept for MIB-based, non-CORBA objects.

7.5. Notes on Initial Design Approach

The Tivoli ORB supports both WAN directory services and WAN object reference/invoke.

Tivoli's ORB is only CORBA 1.2 compliant because they feel that their security interface cannot co-exist with the CORBA V2.0 Internet Inter-ORB Protocol (IIOP). Tivoli is also awaiting standardization of the Common Facilities Transaction Service as they feel that it is critical for network management. Finally, they do not support Context arguments on CORBA interface calls, although this is not essential for the peer-to-peer system.

There are two design approaches for integrating the Tivoli system:

- Wrap Tivoli objects, much like any other non-CORBA objects.
- Attempt to implement the jargons (e.g. MIST) in ORB terms.

The first choice implies that Tivoli is not much better than other choices considered in the survey. Even though it is CORBA-compliant, a CORBA v1.2/CORBA v2.0 wrapper or gateway will have to be provided. On the positive side, a CORBA v1.2/CORBA v2.0 gateway is probably the easiest of the wrapper-based jargons to design and implement.

The second approach means the MIST must be ported to Tivoli and the other jargons must be developed. Tivoli does provide a fairly standard IDL compiler, but there is no mention of persistence or replication in their documentation. The directory service and access are both purportedly WAN-based.

We propose to use the second approach as our initial design; if this approach proves infeasible, the results can be incorporated into the first approach. This approach will give us (and Tivoli) a clear idea of what is required of a CORBA-compliant NMS by third party developers and will give the client an indication of what CORBA/CORBA interactions entail. Eventually, we hope that a Tivoli CORBA/CORBA wrapper would define the API that would run on top of IIOP for Tivoli CORBA V2.

8. Industry Leaders and Products

There are a small number of companies that account for most of the market share in network management tools. The leading contenders and their product lines are:

- IBM/Tivoli/TME10
- HP/OpenView
- Bull/ISM OpenMaster
- Cabletron/Spectrum
- CA/Unicenter
- Sun/Solstice Enterprise Manager

Each of the following sections includes a reference to the company's home page and a reference to the product home page.

8.1. Tivoli TME10

(<http://www.tivoli.com>)

(<http://www.tivoli.com/tivevery/prodhtm.html>)

Tivoli Systems was acquired by IBM in early 1996. The IBM NetView product has been replaced by the Tivoli suite of products.

8.1.1. Environment

The currently supported platforms include:

- Solaris 2.3 or 2.4
- HP/UX 9.0 through 9.05

8.1.2. Framework Services

Tivoli Systems, a subsidiary of IBM, offers a CORBA 1.2 compliant Advanced Developer's Environment. The ORB was developed by Tivoli and is known as the Tivoli Management Framework (TMF). APIs are either developed by linking to C runtime libraries or by linking to client stubs generated by compiling standard CORBA IDL. Currently, only BOA-style stubs are offered.

Tivoli has been active in the OMG, but they claim to be focusing on system management standards, although their submissions can be considered generic approaches to policy definition and propagation, object instance repository/directory services, object instance grouping (known as the Collection Service), and operation scheduling.

Tivoli handles WAN and scalability issues by employing multiple Tivoli Management Region (TMR) servers interconnected with a private protocol. The TMR holds the Tivoli Naming Registry (TNR), an object instance repository, and a directory service.

Tivoli provides extensions to IDL (TEIDL) that support:

- Implementation inheritance from multiple processes across the network.
- Instantiable class, abstract class, meta-class, and mix-in class specification.
- A distributed Security model with the ability to set per method/per object ACLs using both simple TMF and DES algorithms.
- A rich set of method activation policies including shared/unshared server, external, and multi-threaded.
- Initialization and installation features to automate application installation into a live, running system.
- A compiler option for strict OMB/CORBA compliance

Tivoli also provides a number of collateral services (both Common Object Services and Common Facilities Architecture services), particularly a Transaction service, in addition to the Naming and Security services mentioned above.

8.1.3. Event Services

Tivoli provides the Tivoli Event Console (TEC), which is a rule-based correlation and filter engine with a Sybase back-end that communicates with Event Adapters (both Tivoli supplied and custom) and the TMR (which also provides simple thresholding/blocking). Up to six events per second can be handled.

8.1.4. GUI

Tivoli's *Desktop Services* offers X11R5, Motif 1.2, and Windows 3.1-compatible presentation API, but it assumes a request/response model (although callbacks are supported). The main components are:

- Dialog Specification Language/Compiler
- TME desktop display manager and command/callback engine
- Desktop services library
- Gadget library

8.1.5. Security

Kerberosized RPC and an optional data encryption service.

8.1.6. Contacts

This review is based on a presentation and demo given by Tivoli (Lyssa Robinson) at BBN Planet on November 1, on subsequent phone conversations with Lyssa Robinson, on the tutorial manual *Introduction to Tivoli/ADE V3.0*, and on the rest of the ADE Documentation set as specified.

- Contact information for Lyssa Robinson, systems engineer, is as follows: 408-369-3930, 408-369-3939 (fax), lyssa.robinson@tivoli.com, in the Campbell, CA office.

8.1.7. Tivoli References

The following white paper was provided by Tivoli.

8.1.7.1. *Tivoli Management Environment and CORBA*

The purpose of this paper is to discuss Tivoli's perspective with regard to the OMG CORBA specification. Tivoli has been closely involved in the evolution of the OMG specifications within the OMG. Our activities include direct contributions on the evolution of the security specification that is being defined for use within a CORBA environment (the importance of this will be made clear in the sections that follow), the evolution of the CORBA specification itself, and the acceptance within the OMG of the Common Management Facilities specification (derived from the work of the X/Open Systems Management Technical Working Group to define a set of management services based on the CORBA specification). Based both on our direct involvement in a number of the OMG working groups as well as our commitment to base our product on relevant OMG specifications, Tivoli has long been a supporter of the goals of the OMG, and much of its current core technology is based on the work of this group.

Tivoli Management Environment and CORBA 1.2

With the 2.0 release of the Tivoli Management Environment, Tivoli became the first Distributed Systems Management software provider to deliver a solution which is compliant with the OMG CORBA 1.2 specification. The specification has been proposed and accepted as a standard for all common ORB systems. The implementation of this standard as part of the Tivoli Management Framework has enabled customers to utilize TME to manage very large scale heterogeneous computing environments with unprecedented ease.

CORBA 1.2 only specifies the architecture of the ORB model. That is, it specifies the provision of the ORB interfaces but it does not specify the ORB implementation. The ORB could for example, be implemented as part of the operating system or as a stand alone process. The TME framework services provide a server-based ORB, which means that the TME ORB is a continually running program, separate from the operating system. The TME ORB communicates with both the server and the client through separate stubs and skeletons via an inter-process communication facility. A secure remote procedure call (RPC) use to invoke operations on remote objects provides both principal authentication and authorization. CORBA 1.2 also does not specify an interoperability protocol, thus there is no specification that an ORB developer can implement to with any confidence that one implementation can inter-operate with another.

The Tivoli framework services support all the major CORBA 1.2 components including an Object Request Broker, a Basic Object Adapter, an extended IDL compiler with both ANSI C and bourne shell language bindings, an interface repository and the interfaces required for a Dynamic Invocation Interface (DII). The TME framework services do not, however, currently support optional context objects. The CORBA 1.2 specification of context objects is obscure; and there is significant disagreement as to its use and benefit in the standard.

In addition to these components the Tivoli Management Framework supports additional services required for secure and scaleable distributed systems management

solutions. These services provide support for security and transactions at the ORB layer. For example, each ORB is fully authenticated from a central security server, so that an ORB running on one Tivoli managed node will only accept ORB requests from some other ORB in the Tivoli Managed Region (within the security domain of the same central security server). ORB communication can optionally be encrypted to preserve data integrity and privacy. Finally, ORB requests occur within the security context of the administrator making the management request, and this information is used to verify that an administrator has the necessary security credentials to perform the operation (i.e. invoke the requested method on the specified object) in the TMR.

Tivoli also includes support for transactions, which allow method invocations to occur within a top-level or nested transaction context. Within this context, any modifications to any object attribute data is staged for a later commit or abort operation. This mechanism guarantees that persistent object attribute data from objects that reside on more than one ORB can be maintained in a consistent state. Without this capability the recovery actions that must be provided when distributed operations are not able to complete successfully in the distributed environment would be extremely cumbersome, expensive, and error-prone.

What's new in CORBA 2.0?

Unlike CORBA 1.2, CORBA 2.0 is actually a family of specifications, and support a number of different capabilities. The set of specifications include core capabilities, support for interoperability, support for a C++ binding, and support for enhanced portability.

The most interesting extension to the core capabilities offered by CORBA 2.0 is a Dynamic Server Interface. Of course, the most visible CORBA 2.0 specification is the support for interoperability, defined by the Internet InterOperability Protocol (IIOP), which defines the wire-level protocol that allows an IIOP-compliant ORB provided by one vendor to inter-operate with a second IIOP-compliant ORB provided by another. The C++ binding defined by CORBA 2.0 allows a CORBA 2.0-compliant IDL compiler to emit interface definitions using C++ instead of C, so that the C++ class library support built into most C++ development environments can be used on IDL-defined interface definitions. Finally, the enhanced portability services provides additional object life cycle services that increase the chances that interfaces defined by one IDL compiler can be ported to another.

Of these, perhaps the only CORBA 2.0 specification that's particularly useful to DSM solutions would be the support for interoperability. The remainder of the CORBA 2.0 services are primarily of interest to general-purpose ORB vendors, because they increase the likelihood that an application developed on one ORB from one vendor would easily port to a second ORB from another vendor. Since this is not the environment that Tivoli's ORB is used in, the particular value of CORBA 2.0 to Tivoli's customers is more as an indicator of Tivoli's commitment to CORBA-based standards initiatives and to Tivoli's commitment to continue to invest and support its ORB-based technology.

Tivoli's position with regard to CORBA 2.0

As noted above, two base services essential for DSM is support for a fully secure environment and the error recovery inherent in a distributed transaction environment. CORBA 2.0 addresses neither of these two key services, and represents the biggest stumbling block for Tivoli moving to CORBA 2.0. To fully explain Tivoli's position with regard to CORBA 2.0, it's important to address both sets of issues above (the specifics of CORBA 2.0 support AND Tivoli's long-term commitment to CORBA). Let's address the second issue first.

When Tivoli first set out to develop a framework, one of the key requirements was an object-based technology that could cleanly and portably run across hardware and operating systems from multiple vendors. The only technology which meets these requirements is CORBA. Therefore, Tivoli adopted the then-current version of CORBA (1.2) as its base standard both for its ORB implementation and as the programming environment for both its management services and its management applications. Of course, that version of the CORBA specification did not include support for security, transactions, implementation, etc., and Tivoli's implementation of CORBA 1.2 is an "extended implementation" in that it includes support for these services.

When CORBA 2.0 was released, Tivoli evaluated the various additional services included in CORBA 2.0 and determined that the core, C++, and enhanced portability services of CORBA 2.0 did not add sufficient additional benefit to Tivoli to merit the development expense of evolving to the new version of the specification. The one feature that did, on the surface, merit that expense was support for interoperability through the IIOP. However, there was a catch. As noted earlier, Tivoli had to extend the CORBA 1.2 specification to include support for security and transactions, based on our strongly-held belief that these services were essential to a robust, enterprise-wide, scalable systems management framework. While CORBA 2.0 provides a number of useful additions to CORBA 1.2, it does not yet include support for these two essential services. If Tivoli were to embrace the CORBA 2.0 specification, Tivoli would (again) have to extend the implementation of its ORB to provide support for these services. Based on this observation, the benefit of interORB interoperability would be lost once the additional support for security and transactions was added back in (since no other IIOP-compliant ORB would understand how to talk to Tivoli's ORB and visa-versa). So, from Tivoli's perspective, the CORBA 2.0 specification is incomplete, doesn't provide sufficient benefit for the effort involved, and doesn't justify evolving its entire customer base over to a new version of our ORB for such minimal benefit.

What about interoperability?

In many cases customers express interest in being able to inter-operate between Tivoli's ORB and an ORB from some other vendor. This is often the driving force behind questions regarding our plans with CORBA 2.0. As should be clear from statements above, even if Tivoli were to make TME CORBA 2.0-compliant, our security extensions would prevent another, external ORB from communicating with a TME ORB and visa-versa. In effect, the Tivoli security services perform the same function that a firewall performs in a network connected to the Internet.

While IP, in general, provides for cross-network interoperability, the firewall prevents unauthorized access to nodes within the firewall for those users that lack the appropriate permissions. Therefore, CORBA 2.0 compliance would not address the real customer requirement: to be able to have applications developed using one ORB technology access objects within the TME and visa-versa. For those customers with this requirement, there is another approach.

Our recommendation for such interoperability is to define gateway objects that provide interface definitions accessible from within one ORB environment that can then make method calls to analogous objects in the other. In this way, the gateway object can be given the necessary security context to properly invoke the necessary operations within the TME environment and send the results back.

Likewise, a similar gateway object can be defined that intercepts method calls to TME objects in the TME environment and then makes the analogous methods calls to objects in the non-TME ORB environment and forwards the results of these calls back to the caller within the TME environment. This is an approach that maintains the necessary security and transaction context within TME yet provides whatever level of interoperability the application needs. A number of customers with this requirement have used this approach with satisfactory results.

What's the next step?

The OMG is currently working on both a security specification and a transaction service for inclusion in a future version of the CORBA specification. When that future version of the CORBA specification is available, Tivoli is committed to evolving to that (more complete) version of CORBA. Tivoli has been and will continue to be a driving force and significant supporter of the work on the CORBA specification. Today Tivoli is delivering distributed systems management solutions that are based on a subset of the CORBA 2.0 specification and in the cases detailed above we have gone beyond this specification with our extensions for security and transactions services. It is a combination the these services and the CORBA base specification which have proven to be instrumental in successfully solving the problems of distributed systems management.

8.1.7.2. Interview with Tivoli CEO, Frank Moss

In an interview in *Computer Reseller News*, July 8, 1996, n691 p128(1), Frank Moss, Tivoli CEO, had the following comment about the future of network-computing management:

"But we believe that the ultimate direction of network-computing management is application management, whereby customers manage their entire environments from the point of view of the applications they are trying to deploy. After all, it's the applications that are really the heart of their business. We've created an interface specification called AMS-for applications management specifications-which connects TME with applications and application-development environments."

This is a trend that has been observed in looking at several network management tools, namely an emerging emphasis on application-to-application level management in an effort to address the question: "how are my applications running?" In order to answer this

question, management tools must merge information over a wide range of devices operating at different points in the network/system hierarchy, addressing everything from "are my transmission lines operational?" or "are my routers congested?" to "does my system have enough storage?" or "is my system CPU-bound?". This spans some very complex management issues (network management alone has certainly not been "resolved") and it is unclear whether companies can deliver on their goals in this area.

8.1.7.3. Interview with Tivoli CEO, Frank Moss

In an interview published in *InfoWorld*, March 25, 1996 v18 n13 p45(2), Frank Moss was questioned about whether Tivoli would maintain its independence after being acquired by IBM. Moss pledged that Tivoli would remain "cross platform and agnostic about operating systems" and that Tivoli would continue "to work with IBM's databases and platforms exactly the same way we work with other vendors now" (i.e. before the merger).

Given that the merger has been relatively recent (February, 96), it remains to be seen whether Tivoli continues to maintain its platform independence.

8.2. HP OpenView

(<http://www.hp.com>)

(<http://www.hp.com/nsmd/ov/main.html>)

HP OpenView is the leader in market share in network management tools. As specified in the Peer-to-Peer Information Systems Statement of Work, HP OpenView has been chosen as one of the network management platforms that will be used in the peer-to-peer prototype system.

8.2.1. Environment

The supported platforms are:

- HP-UX 9.x and 10.x

8.2.2. Framework Services

Openview is an open, proprietary, object-oriented framework whose objects are strictly Managed Object instances. An object's definition is a combination of its map and other graphic representation information and its SNMP MIB definitions. Integration is seen mainly as the integration of Managed Object Agents and a set of independent applications driven by data collected from the Agents.

8.2.3. Event Services

SNMP traps support.

8.2.4. GUI

OSF/Motif.

8.2.5. Security

User ID rudimentary access control.

8.2.6. Contacts

This review is based on the *HP OpenView Windows Application Design and Style Guide*, *Programmer's Guide and Reference*, *Programmer's Guide* (a separate manual emphasizing SNMP concepts), *Integration Utilities* and the OpenView Network Node Management web pages on the Developer's Toolkits and Integration Certifications Requirements.

- Ginny Johnson 617-221-5286
- Magali Medina 617-238-8508

8.2.7. HP References

8.2.7.1. *HP OpenView Users' Forum Survey*

From *INTERNETWORK* February 1996 v7 n2 p11(1):

A survey of members of the HP OpenView Users' Forum reveals the high degree of frustration of network managers over the shoddy customer support provided by HP. About half of the respondents, however, said that they will stay committed to the product for one to three years for various reasons, such as the need to recoup investments and skepticism about whether there is a better product. The survey also noted what network managers really want. These include a scalable and distributed architecture, built-in reporting capabilities, better event handling functions and more controllable auto discovery, improved SNMP polling features, easy to install products, and a truly open network management system.

The complaints I'm talking about aren't isolated -- the horror stories are consistent, told over and over again by an overwhelming majority of respondents. And our survey reached over one-fourth of the OpenView Forum membership, meaning that the results truly reflect the wider population.

What do these managers want besides better customer support?

* First and foremost, a scalable solution with a distributed architecture -- a product that doesn't get more and more complicated to use when there are multiple users and multiple maps.

* Second, they want built-in reporting capabilities so they can make sense out of the data they've collected without spending months writing scripts and feeding data into third-party reporting tools.

* Third on the list are better event handling capabilities -- filtering, consolidating and correlating alarms. In addition, managers want auto discovery that is more controllable, SNMP polling features that work better, and products that are easier to install.

* Last, but not least, they'd like an open management system that is truly open -- supporting the ability to easily write to an event log and trigger events from a log -- and a common database (a real relational database) instead of several binary flat files.

Is this too much to ask? Interestingly, not one respondent asked for some pie-in-the-sky object-oriented system that supports CMIP and GDMO modeling...

No one asked for the moon or a product that handles everything from soup to nuts. Just simple, reliable SNMP data collection, with information stored in an accessible place, and a client-server architecture that can be distributed in a scalable fashion. Is that really too much to ask?

8.3. Bull ISM/OpenMaster

(<http://www.bull.com>)

(<http://w3ibt.bull.fr/ism/>)

Bull is a recent arrival in the U.S. market, but is strong in European and Asian markets.

8.3.1. Framework Services

Integration is seen as a matter of defining managed objects in ISO's OSI GDMO. There is a client/server architecture, but communication with components is CMIP-based.

8.3.2. Event Services

Supports CMIP events.

8.3.3. Contacts

This review is based on a presentation given by Bull representatives Albert Fitussi and Michael Maloof, and on the Integrated System Management Administrator's Guide.

Contact information is as follows:

Michael Maloof (508.294.6129, MMaloof@bull.com)

Albert Fitussi (508.294.5048, a.fitussi@bull.com)

8.3.4. Bull References

8.3.4.1. Comments from Network World

From *Network World* April 15, 1996 v13 n16 p19(1):

ISM/OpenMaster is a late arrival in the US market, although it is used by 500 organizations worldwide; it competes with HP's OpenView, IBM's SystemView and Tivoli, and Computer Associates' CA-Unicenter.

8.3.4.2. Bull/Amdahl Agreement

From *CommunicationsWeek* March 18, 1996 n601 p38(1):

Abstract

Groupe Bull and Amdahl recently reached partnership agreement that will dramatically expand the market for Bull's ISM/OpenMaster enterprise management software suite, while completing Amdahl's product offerings. The partnership provides Amdahl with a complete solution, enabling it to compete with recently merged IBM and Tivoli Systems. The mergers highlight the necessity of a strong partnership strategy to remain competitive in the network market. However, the trend toward partnership raises interoperability issues. Network managers cite product interoperability as the most important consideration in purchasing distributed systems management tools. Many network managers have not yet deployed a framework for distributed systems

management, even though over half of the network managers responding to a recent survey reported that they would choose HP's OpenView. A clear winner in the network market is yet to emerge.

Full Text

Two weeks ago, I received a call from a PR person who wanted to set up a time for me to speak with executives from Amdahl Corp. about its distributed systems management strategy. My first reaction was: "What distributed management strategy?" I wasn't aware that the company that made a name for itself by giving IBM a run for its money in the mainframe market had a strategy. It does.

The foundation of that strategy is a five-year, renewable partnership with Groupe Bull to market the Paris computer vendor's ISM/OpenMaster enterprise management software suite worldwide. The pact makes sense for both companies. Bull gains access to Amdahl's customers on this side of the Atlantic as well as its sales force.

Amdahl, concerned about what type of an impact an IBM-Tivoli union is likely to have on its customer base, now has a comprehensive solution that manages PCs, work groups, Unix systems, mainframes, data and voice networks as well as enterprise-wide security and database systems.

The Amdahl-Bull deal, following on the heels of the IBM-Tivoli merger, further exemplifies that no vendor can survive in today's complex, distributed networking world without a strong partnership program or strategy.

But as companies form partnerships, network managers are increasingly concerned with how well the products interoperate with each other. A recent survey of more than 700 IS/network managers conducted by Boole & Babbage revealed that most important criterion for evaluating distributed systems management products is interoperability.

The level of interoperability between products from different vendors is the most important criterion for buying DSM tools, said Saverio Merlo, senior vice president of marketing at Boole & Babbage, a provider of management software. The second most important is scalability and the third is vendor support.

Another important survey finding was relatively few respondents had deployed a framework for distributed systems management in their organizations. Deploy means actually using that platform throughout an organization.

When the managers were asked which framework they had chosen, not deployed, 54 percent said Hewlett-Packard's HP OpenView would be their building block.

Though it might seem that OpenView, the market leader in the enterprise management arena, won the contest, the race is not over. OpenView has a commanding lead, but indications are that the fastest-growing net management platform is IBM's NetView for AIX. It remains to be seen what type of impact IBM-Tivoli will have on the distributed systems management market. And others are jockeying for position.

8.4. Cabletron Spectrum

(<http://www.cabletron.com/>)

(<http://www.cai.com/products/uctr.htm>)

Cabletron's Spectrum product has been widely acclaimed in industry publications for the intelligence of its network trouble shooting strategy. Cabletron has put particular emphasis on the problem of isolating the root cause of a network problem, such as a failed line or router, from the hundreds or thousands of related failure reports that a single point failure may cause. However, as Cabletron is a major provider of network hardware, such as routers, it is more focused on providing network management tools for its own hardware, than on providing a truly open architecture suitable for supporting third party applications.

8.4.1. Environment

Spectrum is available on at least the following platforms:

- HP/UX 10.01 on HP9000
- Solaris 2.4/2.5 on Sparc

8.4.2. Framework Services

The SPECTRUM system is officially unbundled to provide users and developers with components and toolkits. The two primary components of the SPECTRUM product are the SpectroSERVER, which communicates with the managed devices, and the SpectroGRAPH client, which provides the user interface graphical display functions. The SpectroSERVER contains a modelling engine, called the Virtual Network Machine (VNM), an object-oriented database (OODB) and a multi-product Device Communication Manager (DCM) for communicating with managed devices.

The SpectroSERVER communicates with the SpectroGRAPH client via the SpectroSERVER API (SSAPI). The SSAPI is implemented in a separately available toolkit which provides third-party applications with an interface to the data maintained in the SPECTRUM database and managed devices. In order to employ a model and export it, the developer must register hardware-address-founded, MIB-based *model definitions* with Cabletron. The model definition is similar to the CORBA interface definition and database schema combined. It is used to communicate management information internally and to make it persistent by storing it in a Sybase-based database. There is a Model Type Editor toolkit available.

The SpectroServer API is a published, but proprietary interface. The API is not CORBA compliant, but Cabletron expects to release a CORBA-compliant API within the year.

The Device Communication Manager, which is part of the SpectroSERVER, is the component that communicates with managed devices. It supports GETs and SETs via SNMP, RPC, and IPC. In addition, there is an External Protocol Interface (EPI) to the Device Communications Manager that can be used to write applications which communicate with devices through non-standard protocols. Editors and toolkits are available to facilitate building access to new network components.

The interfaces to the rule-based configuration and topology engine in the VNM include the Inference Handler API and the Command Line [User] Interface which allows simple rule definitions.

There is no replication of model data between VNMs, although the alarms triggered by events are distributed via the proprietary Distributed Data Manager (DDM). This manager combines a 20:1 compression ratio with TCP/IP transport to ensure that all SpectroGRAPH clients are notified with minimum impact on the network.

The heart of the VNM is the NetMap-like Virtual Network Machine in which the components are connected according to defined and inferred relationships based on hardware addresses. VNM topologies can be saved as an open, proprietary info-base format, which is the only import format. The topologies are also exportable in a number of formats including ASCII files, Sybase, and Oracle.

Configuration fields in a MIB, such as the configuration fields used by routers, can be defined in a special Configuration Manager package and both saved and downloaded to managed systems.

8.4.3. Interoperability with HP OpenView

Interoperability with HP OpenView is achieved via a Spectrum Element Manager (SPEL) Gateway, which will be available as a product at the end of December 1997. However, Cabletron does not intend to support this product.

8.4.4. Event Services

Events and Traps are seen as symptoms and are not propagated by the VNM framework, unless they pass through the primary filters, at which point they are considered Alarms. This event handling is part of Spectrum's efforts to isolate network failures and minimize reporting of multiple events and traps due to a single point of failure. Events are easily exported from the VNM, but importing events is more difficult. Two approaches would be to declare events as primitive alarms or to develop a C++ importing wrapper to map the events into a model definition in the VNM.

8.4.5. GUI

The SpectroGraph client is an Xwindows/Motif compliant GUI. There is also a command line interface.

8.4.6. Security

According to the Spectrum Administrator's Reference Manual, security in Spectrum is primarily an access control/authorization scheme similar to the SNMP V2 concept. It requires additional crypto and network security services to provide authentication, privacy, and data integrity.

8.4.7. Contacts

This review is based on a presentation and demo given by Cabletron on November 6, 1996, the *Configuration Guide for Spectrum Enterprise Management 4.0Rev0, SPMAs 2.1, Spectrum Element Manager 1.01 and StackView 1.0* [June 1, 1996] and other documentation as noted.

- Matthew Palis (603.337.2145)
- Larry Benson
- Carmelita Lawrence (clawrence@ccmailpc.ctrn.com, 603.337.1489) - visit coordinator

8.5. Computer Associates International Inc. CA-Unicenter

(<http://www.cai.com>)

(<http://www.cai.com/products/uctr.htm>)

CA-Unicenter is the second largest computer software company in the world (from *Software Magazine*, April, 1996). Digital Equipment Corporation recently sold their Polycenter technology for network management to CA-Unicenter.

8.5.1. Environment

CA-Unicenter is supported on at least the following platforms:

- HP-UX 9.x and 10.x
- Solaris 2.3, 2.4 and 2.5

8.5.2. Framework Services

Interoperability with OpenView is achieved by including a licensed copy of the OV Operations Center and the SNMP Platform, or for CAI's TNG (The Next Generation), using the Software Developer's Kit (SDK), and using the World View API. Objects are defined in the proprietary Common Object Repository of the Enterprise (CORE) via the Class Wizard tool and the Enterprise Management API (which has multiprotocol data transport).

TNG is not CORBA compliant although it claims to be open and CORBA is listed under the *Adoption of Standards* on the CA-Unicenter feature checklist. Application developers have to use the World View API to define interfaces, not IDL, according to the CA-Unicenter TNG paper by Dr. Elizabeth Nichols.

8.5.3. Event Services

The Event Manager is central event console with distributed, filtered input events. A proprietary Common Communications Interface is used to transport the events to the Event Manager.

The OV Intelligent Agent can send events to the Event Manager.

8.5.4. GUI

OSF/Motif for current version.

Virtual reality business process simulation/model for TNG.

8.5.5. Security

Kerberos API, plus authorization and ACLs; complies with National Computer Security Center Green Book rules.

8.5.6. Contacts

This review is based on discussions with Claudia Peters, the Faulkner Information Services report prepared for CAI, the CA-Unicenter TNG Software Development Kit brochure, and other marketing brochures as noted (received from Rick Schrader).

- Claudia Peters (617.251-5538) is the technical contact.

- Rick Schrader (703.709.4762) is the sales rep for Army related projects.
- John Petracek (703.709.4590) is our new sales rep and the sales rep for Air Force related projects. petracek@mnsinc.com
- Kyle Hodges (516.342.2376) is the original marketing contact assigned to us.

8.5.7. CA References

Digital's POLYCENTER technology has been acquired by CA-Unicenter

CA-Unicenter will support Digital's Polycenter customers with a probable migration to CA-Unicenter TNG (The Next Generation).

From *LAN Magazine* , August 1996, v11 n8 p18(1):

Computer Associates (CA, Islandia, NY) has made it official: Its eagerly awaited CA-Unicenter TNG is shipping and will be hitting the market six months ahead of the scheduled ship date of early 1997...

This successor to CA's enormously popular Unicenter product includes many new features and tools that provide management functionality for every enterprise network resource, including databases and applications. "This is the only product that focuses on managing processes from end-to-end--mainframe to Internet and back" Kumar says.

The product does this managing through Jasmine, an architecture based on CA's object-oriented database for multimedia applications. Through this model, information about resources can be stored and monitored, allowing administrators to keep better track of widely distributed systems. This architecture also provides an open, extensible environment that is highly scalable and customizable.

Because of its single, centralized repository, Unicenter-TNG manages everything from one console, regardless of the platform. It also supports the major network management products, including HP OpenView, Sun Net Manager, and Cabletron's Spectrum.

The product also includes a three-dimensional user interface, creating an intuitive view of system resources, including hardware, software, security, databases, and Internet resources.

8.6. Sun Solstice Enterprise Manager

(<http://www.sun.com>)

(<http://www.sun.com/solstice/em-products/network/ent.man.html>)

Solstice Enterprise Manager Version 2.0 superceeds and incorporates the earlier SunNet Manager, Site Manager and Domain Manager. The Solstice Enterprise Manager can be unbundled into server and client (user interface) components. The server component supports third party applications through the portable management interface (PMI) which is based on CMIP over TCP/IP. Sun plans to provide a Java management API to their server by next spring, but they have no plans for supporting a CORBA interface for third party applications.

8.6.1. Environment

The supported platforms are:

- Solaris 2.4 or 2.5

8.6.2. Framework Services

Solstice Enterprise Manager is based on technology acquired from Netlabs, Inc. The Enterprise Manager includes server and client components. The server component communicates with the managed devices and with the client component; the client component provides the user interface. Third party applications can interface to the server using the same protocol as is used by the Solstice Enterprise client and server components. This interface uses CMIP over TCP (CMOT) based and is not CORBA compliant. A Java interface should be available in Q3 '97, but there are no plans to support a CORBA interface to the server. The Enterprise Manager includes proprietary object definition interfaces and installation tools (GDMO compiler, RPC CMOT transport). The Enterprise Manager is Omnipoint compliant.

The instance repository is hierarchically structured (MOIs are assigned to a manager who in turn can be assigned to a manager). However two Management Information Systems (MIS) can act as reciprocal managers/agents for each other.

The API (for third party applications) is GDMO/CMIS based.

8.6.3. Event Services

The RequestDesigner (part of the Nerve Center) is a tool used to define an event filter and assign an action to be launched on match.

8.6.4. GUI

The Enterprise Manager includes a Motif-compliant user interface.

8.6.5. Contacts

This review is based on a presentation by and discussion with Tom Bialaski and Bob Ganley on 12 Nov 1996 and on the *What's new in: Solstice Enterprise Manager 1.2 Developer's/OEM Release* notes. [Feb 29, 1996]

Contact information is as follows:

Tom Bialaski, Systems Engineer (508.442.0577), tom.bialaski@east.sun.com

Bob Ganley, Senior Account Manager (508.442.0352), robert.ganley@east.sun.com

8.6.6. Sun References

8.6.6.1. Sun Solstice Enterprise Manager

(<http://www.sun.com/solstice/em-products/network/ent.man.html>)

Solstice Enterprise Manager shares common technology with SolsticeTM Domain ManagerTM, SolsticeTM Site ManagerTM, and SolsticeTM SunNet ManagerTM. Applications and agents written to the Solstice SunNet Manager (SNM) APIs will run on Solstice Enterprise Manager. With SolsticeTM Cooperative ConsolesTM receiver included, it can receive management information sent by Solstice Domain Manager, Solstice Site Manager, or Solstice SunNet Manager thereby enabling it to act as a manager of managers in a network management hierarchy.

9. Innovative COTS Systems

In addition to the market share leaders, we have researched companies with smaller market share, but innovative solutions, notably SMARTS InCharge.

(<http://www.smarts.com/company.html>)

(http://www.smarts.com/products/incharge_datasheet.html)

InCharge is one of the few management platforms that can efficiently correlate large numbers of symptoms into probable causes. InCharge uses a patented algorithm based on coding theory to lookup problem causes based on the current set of symptoms. This gives a (claimed) performance improvement of a factor of 100 over other systems, i.e. 1000 symptoms/sec versus 10 symptoms/sec (Tivoli).

The InCharge platform works as follows: Raw information is collected from devices using SNMP, CORBA, or a user supplied interface. The raw data is compared against thresholds to generate symptoms. All the symptoms are input into the correlating engine to determine any problems. Problems can be sent up a hierarchy of InCharge platforms to the ultimate Network management applications, such as a trouble ticket or network map application.

BBN is using SMARTS InCharge as part of a Related DARPA project, DIRM.

10. Standards Organizations

10.1. Distributed Management(disman) group of the IETF (Internet Engineering Task Force)

(<http://www.ietf.org/html.charters/disman-charter.html>)

10.2. The IEEE Communications Society (IEEE ComSoc)

(<http://engine.ieee.org/comsoc/>)

10.3. The International Federation for Information Processing (IFIP)

(<http://www.ifip.or.at/>)

10.4. EWOS

(<http://www.ewos.be/index.htm>)

EWOS is a European forum of users, manufacturers, implementors and procurers, working to achieve openness in Information and Communications Technologies. They publish a Guide to Open Systems Specifications: Systems and Network Management Technologies (<http://www.ewos.be/nm/gmtech.htm>).

10.5. Network Management Forum

(<http://www.nmf.org>)

NMF exists to promote and accelerate the worldwide acceptance and implementation of a common, service-based approach to the management of networked information systems. A non-profit corporation, NMF is funded by its members, including organizations that consume information and telecommunications services, organizations that provide networked services, and organizations whose telecommunications and computing products are used to create services.

The Network Management Forum recently announced that it was realigning its work programs in this press release.

NMF Realigns Work Programs to Match Investment Trends

All deliverables to carry NMF brand name

Barcelona, Spain, October 29, 1996. NMF today announced a restructuring of its work in order to remain closely aligned with the investment priorities of service providers and product developers. Work programs will now be grouped into three focused areas: service management, network management and platforms & technology. The agreements, specifications and outputs from NMF have also been restructured, to permit faster publication to the industry.

"Part of NMF's success has always been our willingness to regularly tune programs as market requirements evolve and to quickly adapt as competitive pressures impact members' priorities," commented Keith Willetts, NMF President. "Although we have been working in all three areas for some time, they have not always had equal priority. These changes should significantly improve NMF's effectiveness and make it easier for developers and implementors to find the integrated solutions they require."

Recent focus groups conducted by NMF indicated three major areas where service providers are making significant investment in management systems:

- Improving process flow-through to achieve service management excellence;
- Managing an enlarging network infrastructure to meet rising demand for bandwidth,
- Migrating from proprietary, legacy systems to integrated, "off-the-shelf" distributed computing platforms.

In the area of Service Management, NMF will continue to play a lead role, as it has since introducing its SMART program two years ago. Based on this work, service providers have identified areas where industry agreements are needed to achieve process flow-through -- particularly in areas such as global alliances and Internet Service Providers. Six teams are active in the service management area, addressing

issues such as trouble ticketing, performance reporting, ordering, service configuration, and billing.

Network management, the starting point of NMF in 1988, has continued to evolve. Initially the main driver for network management standards was supplier independence but today, the ability to manage efficiently across multiple networking technologies is the critical requirement. Work done by other groups, to address the management of ATM, Frame Relay and SDH/SONET, has to be integrated in order for service providers to achieve the service flexibility they need to stay competitive.

In its Platforms & Technology program, NMF will merge two parallel activities: the definition of distributed computing platforms (known as SPIRIT), and the definition of management middleware specifications, including interworking specifications and development tools. By bringing these together, NMF members should be better able to apply "off-the-shelf" technology to management systems.

With clarification of its work programs, NMF is also making changes to the way it publishes its agreements and specifications, and the names they are given. In the past, outputs bore the name of the program that produced them (such as OMNIPoint), all deliverables will now be marketed under the NMF name. Each document will also be clearly labeled as to the subject matter that it covers.

Three types of deliverables will come from NMF's Service Management and Network Management programs: NMF Requirements specifications, NMF Design and Analysis specifications, and NMF Solution Sets. Requirements documents, which will now be made available to non-members, spell out the business agreements for exchanging management information. Design and Analysis documents, detail an implementation-neutral Information Model that can be implemented in multiple standards environments. Solution Sets, introduced in 1995, capture implementation-specific technical interface specifications, removing all options and virtually guaranteeing interoperability between implementing systems. Six Solution Sets (released under the OMNIPoint name in 1995) have been introduced with more expected in early 1997.

Two types of deliverables will come from NMF's Platforms & Technology program. The SPIRIT specification, which is jointly sponsored by NMF and X/Open, will retain its name, and will continue to be updated as service providers endorse the use of market-available computing standards. NMF Component Sets, also introduced in 1995, provide detailed management middleware specifications. Four Component Sets (also released under the OMNIPoint name) are already available, and two more are expected to complete shortly.

"Taken together, these changes will enable us to provide highly specific solutions in areas where the industry requires a standards-based solution while providing a clear and consistent name to the market," explained Elizabeth Adams, NMF Managing Director.

NMF is a non-profit, global consortium providing the communications industry with a forum for developing open, integrated solutions for managing large, complex networks. NMF offers a range of programs, publications and specifications that make it easier for

companies to reduce costs, improve service quality and introduce new products and services faster.

Over 180 of the world's leading service providers, equipment suppliers, software developers and private network operators from 28 countries work together as NMF members.

11. Vendors' user groups and partnerships

11.1. HP OpenView Forum

(<http://www.hp.com/nsmd/ov/main.html>)

On an annual basis, OpenView Forum collects members' product requirements, asks members to rank the priority of each requirement, and forwards the requirements and rankings to appropriate HP OpenView technology providers for their response.

11.2. Tivoli 10/Plus Association

(<http://www.tivoli.com/TENPLUS>)

We believe the 10/Plus Association represents a watershed in the management industry. All too often projects fail to reach their full potential due to the lack of a comprehensive enterprise management solution. The problem is, no single vendor--not even the combination of IBM and Tivoli--can solve all of your enterprise networking computing needs.

By working smarter, and making a major investment in cooperatively creating products and standards with our customers, Tivoli Systems and our 10/Plus partners are focused on changing the way vendors work together. These industry-leading partners are working jointly with Tivoli to create a sophisticated level of integration with our network computing product suite, and comprehensive support programs, so that there can be no doubt that the 10/Plus Association is at the forefront of solving the real customer problem of today--lack of integration.

12. Conferences

The IFIP and IEEE hold three series of conferences:

- IM, held in odd years, the next will be in May 1997,
- NOMS, Network Operations and Management, held in even years, and
- DSOM, Distributed Systems Operation and Management, a small workshop group which meets annually, most recently in October 1996

12.1. IM '97 The Fifth IFIP/IEE International Symposium on Integrated Network Management

(<http://engine.ieee.org/comsoc/IM/>)

"Integrated Management in a Virtual World"

May 12 - 16, 1997

... the fifth biennial IFIP/IEEE International Symposium on Integrated Network Management. The theme for 1997 is Integrated Management in a Virtual World, focusing on the pivotal role that integrated network management plays in worldwide information networks and distributed systems that cross geographical and political boundaries. Indeed, these networks extend beyond physical boundaries to support virtual corporations, virtual LANs, inter-enterprise internetworking, real and virtual service management, outsourcing and electronic commerce. This premier Symposium continues the highly-regarded series sponsored by IFIP Working Group 6.6 on Network Management and the IEEE Communications Society Committee on Network Operations and Management.... This program addresses the increasing interest in overall management solutions across all types of networks, enterprise communication systems, distributed computing systems and applications.

History of the IFIP/IEEE Symposium on Integrated Network Management (IM)

Known by the acronym "ISINM" until this year's change to "IM," which is much easier to say, this continues to be the premier biannual industry event for Network Managers.

Since 1989, the Symposium has provided a central technical exchange forum for the research, standards, development, systems integrator, vendor and user communities in network management. With the global information infrastructure growing at an exponential rate, IM '97 promises to build on the successes of previous symposia and provide a unique opportunity for exploring integrated management solutions with a diverse international community.

The symposium series has demonstrated increasing interest in overall management solutions across all types of networks, enterprise communication systems, distributed computing systems and applications. Such comprehensive network management is of continued interest for the next symposium: integrating data and telecommunication networks, from narrowband to broadband, terrestrial to satellite, fixed to mobile, used for common as well as advanced multi-media communication.

Beginning with our first symposium in 1989, each ISINM program and its related theme has reflected the historic events in integrated network management, indeed has helped shape them.

1989: Improving Global Communication Through Network Management

When we held the first ISINM in Boston in 1989, the need for comprehensive network management capabilities was apparent after major disasters had occurred in the telecommunications industries in the years before. Standards for enabling integrated network management across multiple vendor networking resources were in the heat of development in international and regional arenas. While some thought that developing these standards was the most difficult path on the road to integrated management solutions, many realized a few years later that standards were the beginning of a long journey. Integrated network management emerged to be one of the most complex and hard to solve problems of our heterogeneous communications community.

1991: Worldwide Advances in Integrated Network Management

After two years, when we held the second ISINM in Washington, D.C., the need for enterprise-oriented management across data and telecommunications applications and

distributed systems became increasingly apparent. Principle problems related to incorporating standards into products aimed at providing coherent, integrated network management solutions across future, standards-based, multi-vendor components as well as existing proprietary components. Multi-vendor demonstrations in North America, Europe and Japan seemed to indicate that the time had come when users could competitively procure network management products in any of several countries and be confident that they would interoperate with comparable products in other world regions. That wasn't so.

1993: Strategies For The Nineties

We have learned. We are not at the end of the road - we are not even in the middle. We are only at the beginning and will remain there probably for the greater part of the nineties. Worldwide coordinated strategies are needed to evolve integrated network management in the best way. The beginning of the nineties was characterized by big political, ecological and technical changes in all areas worldwide. The exponential growth of internetworking in general and new multimedia applications based on broadband and mobile network technology will remain the driving forces of the communications area.

However, the element of uncertainty plays a dominant role in all environments. Downsizing and up-sizing in volume and time requires flexibility to change. These problems are intensified by economic and regulatory constraints, problem complexity, technology advances, standards development, product introductions, market requirements, user demands and other factors which change unpredictably over time.

A paradigm shift took place during these phases: network management systems used for crisis situations in the past evolved to powerful tools for the day-to-day management of systems, services, applications and, of course networks.

1995: Rightsizing in the Nineties

During this sometimes turbulent period of rightsizing in all areas, the need for management systems is greater than ever before. Management is a fundamental part of a reliable information infrastructure. It assures the correct, efficient and mission-directed behavior of the hardware, software, procedures and people that use and provide all the information services. Effective management of the information infrastructure is becoming as essential as marketing and selling products. In addition, it helps to raise customer satisfaction. Integrated network management belongs to the enabling technologies of a worldwide information infrastructure.

12.2. DSOM, October 28-30, 1996

(<http://www.cselt.stet.it/CNOMWWW/DSOM.html>)

The workshop is sponsored by the International Federation for Information Processing (IFIP) Working Group 6.6 on Network Management for Communication Networks with technical co-sponsorship by the IEEE Communications Society Technical Committee on Network Operations and Management (CNOM).

The workshop is an effort to bring together people actively working in the Distributed Systems Management area. The workshop attendance is limited to 100 participants.

The scope of this workshop will be on the operations and management of application software or services within a distributed system and the impact of advanced computing and network technologies on management.

The "hot topic" theme of this workshop will be on "Intelligent Agent Technology for Management of Distributed Systems" with papers distributed over 3 sessions. Recent trends in distributed processing is to make use of intelligent agents which interpret a scripting language such as TCL or Java which enables sophisticated new management functions to be loaded into existing management agent interpreters. A variation on this concept is the use of mobile agents which move around the network performing actions on behalf of users (managers). As new technologies become available and market pull is getting strong new management paradigms are being proposed and they will be discussed at the Workshop.

The workshop will also explore the problems and issues relating to the use of dynamic programming concepts for intelligent agents in managing distributed systems, but will also provide papers on other topics related to distributed management, such as:

- Experiences with Distributed Management of Applications and Services.
- Coping with Management of Large Scale Distributed Systems
- Cooperative Management
- Quality of Service in Distributed Systems and Networks
- Data Management in Distributed Environments
- Standardization in Distributed Management
- Platforms for Distributed Management
- Management Policies
- Security Issues

13. Research Projects

13.1. *The Simple Group*

(<http://wwwsnmp.cs.utwente.nl/~nm/>)

The Simple Group is the network management research group of the 'Tele-Informatics and Open Systems' (TIOS) group of the 'Centre for Telematics and Information Technology' (CTIT) of the 'University of Twente' (UT) in the Netherlands. They maintain the Scotty (<http://wwwsnmp.cs.utwente.nl/software/ut-scotty.html>) system which is freely available as source code written in C and Tcl/Tk for UNIX systems.

Scotty is the name of a software package which allows to implement site specific network management software using high-level, string-based APIs. The software is based on the Tool Command Language which simplifies the development of portable network management scripts. Scotty was originally developed at the University of Braunschweig.

They also maintain the SimpleWeb (<http://wwwsnmp.cs.utwente.nl/>) which is a listing of network management resources, including a list of network research projects. (<http://wwwsnmp.cs.utwente.nl/research/groups.html>)

Scotty provides access to SNMPv1 and SNMPv2 and a number of well known Internet services like DNS, various ICMP packets, NTP, TCP, UDP, SUN RPCs (mount, rstat, portmap) etc. The tkined network editor is the graphical user interface which integrates applications that are usually written as Tcl scripts based on the scotty Tcl extension. Applications distributed with the scotty and tkined sources include network discovery, trouble-shooting applications, event filter, SNMP MIB browser etc. An experimental MIB browser is also available.

13.2. ARPA ActiveNets Project

The focus of the NetScript, Active Networks and SwitchWare projects is to make it easier to modify the underlying communications nodes. The abstracts for each of these projects cite the time delay (measured in years) for achieving communications protocol standardization, and suggest methods for enhancing communications nodes without need for a long standardization process. These projects are all sponsored by ARPA under the recently launched ActiveNets program.

13.2.1. Columbia University Management by Delegation Project

(<http://www.cs.columbia.edu/~german/mbd.html>)

The agent technology has matured through a few generations of research versions. It was licensed by Columbia to System Management Arts (SMARTS), a DCC lab spin-off company, that recently announced a commercial version. This is the SMARTS company referred to in the "Innovative COTS Products" section above.

13.2.2. Columbia University NetScript Project

(<http://www.cs.columbia.edu/~dasilva/netscript.html>)

Netscript is a programming language and environment for building networked systems. Its programs are organized as mobile agents that are dispatched to remote systems and executed under local or remote control. The goal of NetScript is to simplify the development of networked systems and to enable their remote programming.

13.2.3. MIT Active Networks Project

(<http://www.tns.lcs.mit.edu/activeware>)

A paper (<http://www.tns.lcs.mit.edu/publications/sospwip95.html>) was presented at the 15th Symposium on Operating Systems Principles.

Active networks allow individual user, or groups of users, to inject customized programs into the nodes of the network. "Active" architectures enable a massive increase in the complexity and customization of the computation that is performed within the network, e.g., that is interposed between the communicating end points.

13.2.4. University of Pennsylvania and Bell Communications Research SwitchWare Project

(<http://www.cis.upenn.edu/~jms/SoftSwitch.html>)

We propose the development of a set of technology which will enable rapid development and deployment of new network services. The key insight is that by making the basic network service selectable on a per user (or even per packet) basis, the need for formal standardization is eliminated. Additionally, by making the basic network service programmable, the deployment times, today constrained by capital funding limitations, are tremendously reduced (to the order of software distribution times). Finally, by constructing an advanced, robust programming environment, even the service development time can be reduced.

13.2.5.BBN ActiveNets Project

As part of the ActiveNets project, BBN is working on the SmartPacket project, which applies active network technology specifically to the problem of network management.

13.3. *New ARPA Initiative in Active Networks*

ARPA is initiating a new research program in the Active Networks area. This program will focus on research into making virtual networks robust, secure, and dynamically self-configuring, based on the requirements of the distributed application, and the run-time capabilities of the network resources. Managing the network resources when both their capabilities and capacities are dynamic requires management access that is as low-level, robust, and dynamic as the resource objects themselves. One area of research that BBN is proposing involves a low-level directory service that matches the requirements of distributed processing management, rather than using naming schemes based on e-mail requirements or a simple host name / IP address.

13.4. *Network Management in a Multi-National Environment*

(<http://www.ndf.rl.af.mil/~accord/netman.html>)

This experiment will develop and evaluate emerging network management concepts in a coalition force environment. It will also provide management of the ACCORD testbed network, and support many of the experiments using the network, with management and data collection services.

13.5. *ACCORD project*

(<http://www.ndf.rl.af.mil/~accord/accord.html>)

ACCORD is a five-nation ATM network testbed to enable cooperative demonstration and experimentation of technologies and ideas relevant to military Command Control Communications and Intelligence.

14. Trends and Emerging Technologies

There are several trends and emerging technologies, including:

- Web based management tools as a way around long lead time protocol development (See below.)
- Merger of network and system administration for "end-to-end" monitoring and more emphasis on client or user viewpoint; for example, managing video transfer by either allocating large network pipes or, enabling compression at end points or most interesting, by combination of both

- Management of managers; often done in a "proprietary" way, i.e. one manager provides the ability for other managers to report to it; not a peer-to-peer relationship
- Microsoft Systems Management Server (http://www.microsoft.com/msdn/sdk/platforms/doc/backoff/bpr/src/sms_1.htm) a "de facto" standard for PC management,
- Each manufacturer has both LAN and WAN management products, how will these be integrated in a "manager of managers" scheme?
- Applications are driving network management; the large (i.e. bandwidth consumers) of the internet are web applications, email, manufacturing (CAD/CAM), distributed databases, and video; what are the application users and suppliers in these fields doing?
- Mobile computing
- Satellite communications

Web-Based Enterprise Management

(<http://wbem.freerange.com>)

SAN FRANCISCO - July 17, 1996 - BMC Software Inc., Cisco Systems Inc., Compaq Computer Corp., Intel Corp. and Microsoft Corp. today proposed an industry standards effort that will allow administrators to use any Web browser to manage disparate systems, networks and applications. The intent of the Web-Based Enterprise Management effort is to enable the development of tools that reduce the complexity and costs of enterprise management.

In addition, the effort promotes the use of two new management-related technologies to provide data modeling, manipulation and communication capabilities recently outlined at a meeting of the Internet Engineering Task Force (IETF):

HyperMedia Management Schema (HMMS), an extensible data model representing the managed environment

HyperMedia Management Protocol (HMMP), a communication protocol embodying HMMS, to run over HTTP

15. Industry Comments

15.1. Network Computing Online

(<http://techweb.cmp.com:80/techweb/nc/docs/aboutnwc.html>)

Four of the top 5 vendors (IBM, HP, Sun and Cabletron) were reviewed by Network Computing Online in an August 15 article entitled

Global Network Management -- 4 Platforms Reviewed. Network Computing Online has many corporate sponsors including Cabletron, HP, IBM Networking, and SunSoft and reviews products in a network testbed which includes university and corporate computing centers.

This was a difficult test to call because all of the products are an excellent foundation for some network bias. But Cabletron's Spectrum and HP OpenView NNM proved to be the best. Both have a fine distributed strategy and have added tools and functions to their base platforms. HP is still the leader in open systems support, while Spectrum's offering is the most intelligent. This is not to say the products from IBM and SunSoft failed. Both have workable, distributed strategies, and IBM has some useful tools where the most benefit is derived from a commitment to its hardware and software strategy.

15.2. Communications Week

The following is from an editorial in *CommunicationsWeek* April 8, 1996 n605 p34(1):

Abstract

Distributed services management is a major trend in 1996, with IBM moving to integrate the Tivoli Management Environment (TME) network management platform with its own network management tools after acquiring Tivoli Systems Inc and HP rolling out more pieces of its Tornado distributed-management strategy, such as Network Node Manager 4.1. These and other vendors are moving in the right direction to offer the unified, seamless management solutions increasingly needed on heterogeneous corporate internetworks. Today's management platforms manage networks and systems well but do not give administrators a clear view of the overall effects network problems will have on the enterprise. Cabletron Systems Inc may be closest to a true business-driven enterprise management model because it already has a scalable, distributed architecture in place.

Full Text

If last year was the year of merging network and systems management, 1996 will certainly go down as the year of distributed systems management.

IBM Corp. took the industry by surprise earlier this year with the announcement that it was going to acquire Tivoli Systems Inc. With the merger completed, IBM unfolded the Tivoli road map last week at NetWorld+Interop, positioning the Tivoli Management Environment (TME) as the foundation of its distributed management strategy.

Hewlett-Packard Co. also rolled out another piece of its Tornado distributed management strategy, Network Node Manager 4.1, giving network managers the ability to distribute a number of management functions across their enterprise networks.

And Cabletron Systems Inc., Rochester, N.H., continued to build on its distributed architecture by announcing tighter links with key systems management partners' tools, extending the Spectrum platform's systems-administration reach.

Although much work still needs to be done, these vendors--and others--are moving in the right direction. Network managers are searching for solutions that will help them effectively manage the numerous network devices, mainframes, Unix systems, PCs, servers, databases, applications and public network services that make up today's global networks.

Management platforms are good at managing networks and systems but can't really give IT managers a clear view of what impact a failed router may have on the accounting department or some other business function.

In many companies, the move to distributed applications and object systems has already begun. It's also apparent that network managers are going to require administration tools that do much more than let a sharply defined window into their networks, systems and applications.

Perhaps Cabletron, which already has a scalable, distributed platform, is pointing the way to the future by linking enterprise management with the processes that drive a company's business.

This is a sign of things to come. Network-administration tools must either evolve or be replaced by management software that can see beyond network components--software that can tie into a company's business logic and business processes.

The industry is focused on the problem of performance at the end-to-end applications level (or at an even higher level, on the operation of whole business units). However industry analysts overstate the current state of affairs. The abstract for the above article states: "Today's management platforms manage networks and systems well but do not give administrators a clear view of the overall effects network problems will have on the enterprise."

In fact, there are still major problems in managing networks, (isolating the initial failure, determining what problems are in fact merely symptomatic of another failure, and determining how to fix the problem).

15.3. Gartner Group

(<http://www.cabletron.com/analyst-reports/Gartner-Group/gartner7.html>)

According to the Gartner Group report for June, 1996, the following companies and products accounted for 95% of the market: SunNet Manager, SPECTRUM, NetView, OpenView and ISM/OpenMaster.

SunNet Manager has since been incorporated into the Sun Solstice Enterprise Manager and IBM Net View has been superseded by TME from the new IBM/Tivoli group.

15.4. Network Management Market

In *Software Magazine*, June 1996 v16 n6 p37(7), there are several references to the network management market:

"Tivoli may have a good shot at a market-leading role, but Computer Associates International Inc. and Hewlett-Packard Co. aren't about to give an inch without a good fight."

"For example, Computer Associates International now leads the Unix system management software market with nearly a 21% market share, according to IDC (International Data Corp. Framingham, MA)"

"Furthermore, data from International Data Corp. shows that HP takes the lead when IS shops are asked to name their primary management platform."

16. Web Sites for more information:

16.1. Network Management

<http://smurfland.cit.buffalo.edu/NetMan/>)

This server functions as the archive base for comp.dcom.net-management, as well as for a place to bring together references to other applications and servers. In addition, this site acts as a mirror site for applications, utilities and FAQs pertinent to Network Management.

16.2. SNMP News

(<http://www.int.snmp.com>)

This is a quarterly publication of SNMP Research International. The first publication is 4th quarter, 1996 and covers such topics as Web-based Management, and SNMP v2 Security. This is the publication of a single commercial company, and as such, will undoubtedly reflect their perspective; however, this company has been involved in the research community for many years and newsletters are also likely to cover issues of general interest to the community.

16.3. TechWeb

(<http://www.techweb.com/>)

Articles on technical topics, including network management. Recent articles of interest include the following:

“Enterprise Management Is Just Around the Corner:” October 15, 1995

We tested some preview editions of the next generation of network management products from SunSoft, HP and Cabletron, and the latest shipping version from IBM. Here's what we found...

“Net Admins Demand Cohesive Management”, September 18, 1996

“Problem or Solution? -- Net Management and the 'Net'”, March, 18, 1996

“The Big Picture -- Central control of distributed nets remains a work in progress, but remote monitoring technology is starting to take shape”, July 3, 1995

Section 2 - Peer to Peer Information System Management Architecture

17. Executive Overview

17.1. Identification

This is the *System Architecture Document* for the Peer to Peer Information System Management program. It is submitted in fulfillment of requirement 4.1.3.4 and as input to 4.1.4; and CDRL A006; from the Statement of Work for the Peer-to-Peer Information System Management Contract #F30602-96C-0049.

18. Document Stylistic Conventions

This document is published both in paper and hypertext formats. The cross-references are identified inline, external references are collected in an Appendix, *On-line References*.

18.1. Document Status

This document was updated on 3/27/2002 12:45 PM by lbob@bbn.com (Linsey O'Brien)

18.2. System Overview

The Peer to Peer Information System Management [*P2P*] is designed to integrate the management of a variety of information systems and their associated network components, such as IP and ATM devices. This will be accomplished by integrating selected existing management systems, which will in turn be done by providing optimized communications mechanisms in a standard framework enabling the management systems and their applications to interact as peers. These interactions include managing the management systems and the peer communications components themselves. Provisions for securing the interactions will be outlined in the system architecture but are not considered a primary requirement of the P2P project.

18.3. Document Overview

The purpose of this document is to describe the functionality and architecture of a distributed management system that supports the requirements of the *Peer to Peer Information System Management* program. It will document all the major functionality of the system and describe all the required interfaces, especially the integrated data distribution mechanisms. A brief description of the contents of this document is provided below.

The System Overview describes the high-level functions of the system as they relate to the requirements stated in the Requirements Document.

The Referenced Documents section lists supporting documents referenced directly or indirectly in this specification.

The System Design Decisions section covers the key problems and issues that drove the System Architecture.

The Functional Component Description takes the high-level requirements summarized in the System Overview and System-wide Design Decisions and starts the process of translating them into software design. It first breaks into components the functionality defined by the requirements. It then translates those abstract components into a set of system objects and their interactions.

19. Referenced Documents

19.1. Contractor Specifications

- BBN Systems & Technologies Report 8180, Nov. 1996: Network Management Study
- BBN Systems & Technologies Report 8199, June 1997: Peer to Peer System Requirements
- BBN Systems & Technologies Report 6986 Rev. 3.0, Jan 1993: Introduction to Cronus
- BBN Systems & Technologies, Cronus Documentation, Release 3.0, 1 December, 1992: Cronus Operator Manual©
- BBN Systems & Technologies, Cronus Documentation, Release 3.0, 1 December, 1992: Cronus Operator Manual
- BBN Systems & Technologies, Cronus Documentation, Release 3.0, 1 December, 1992: Cronus User Manual
- BBN Systems & Technologies, Corbus Documentation, Release 2.0, April 1996: Corbus Operator's and Installation Manual

19.2. Other Specifications

- I. Object Management Group publication PTC/96-03-04: CORBA 2.0 Specification
- II. HP OpenView:
 - A. HP OpenView Network Node Manager Products Installation Guide: J1172-90001
 - B. HP OpenView Using Network Node Manager: J1172-90003
 - C. HP OpenView Network Node Manager Reference: J1172-90004
 - D. HP OpenView Integration Series
 - 1. Integration Concepts: J1177-90000
 - 2. OpenView Windows Application Style Guide: J1177-90002
 - 3. OpenView Windows Developer's Guide: J1177-90003
 - 4. OpenView Windows Developer's Reference: J1177-90004

5. SNMP Developer's Guide and Reference: J1177-90005
 6. Integration Utilities: J1177-90006
- III. Tivoli TME (Note: Both the current and previous component names are given in some titles as follows [Current / Former])
- A. Tivoli TME10 Application Extension Facility User's Guide V3.0: GC31-8345-01
 - B. Tivoli [Mid-level Manager / Sentry] Platform User's Guide V3.0: GC31-8323-00
 - C. Tivoli [Framework / Management Platform] Planning and Installation Guide V3.0: GC31-8382-00
 - D. Tivoli [Framework / Management Platform] User's Guide: V3.0 GC31-8322-00
 - E. Tivoli Framework Services Manual V3.0: GC31-8348-01
 - F. Tivoli SNMP Monitoring Collection Reference V3.0: SC31-8388-00
 - G. Tivoli / Enterprise Console Event Adapter Guide / SNMP: SC31-8342-00
 - H. Tivoli / Enterprise Console Event Adapter Guide / HP OpenView: SC31-8338-00

20. System-wide Design Decisions

This section summarizes the System Requirements document and indicates how these requirements affect the system architecture design

20.1. Management Application Integration

The system shall support integrating management information input and output in order to coordinate peer management systems. Such coordinated management is delimited by four functions:

1. monitoring status and other attributes
2. providing historical context for attribute monitoring
3. monitoring events
4. issuing control commands

Integrating two network management systems so that they can perform such functions as peers requires transferring each of the four categories of information about target components to and from the management systems in a peer to peer fashion. Each of the four functional categories of information transfer is implemented as one or more jargons.

20.2. Target Components Integration

The network management functions and information that drive the distribution and security requirements derive from standard ATM and IP components. The ATM resources will be

controlled according to XBind, plus relevant standard SNMP MIB definitions available. The IP resources also will be managed according to their standard SNMP MIB definitions as made available by the network management systems. To the extent that management APIs are defined by standard MIBs, transferring such information between management systems is relatively straightforward, but to the extent that such management APIs are implemented in system-specific object models, transferring information between such systems may require significant inter-system communication integration. In short the application layer is compatible, but the underlying distribution infrastructure is not.

20.3. Object-Oriented Architecture

The system shall support object-based datatypes and provide object methods so as to support access to management information from a wide variety of network components and management applications without having to explicitly specify each individual interface.

This requirement is derived from the need to integrate peer-managed target managed objects into systems whose models tend to be hierarchical. Object-oriented systems' object instances are inherently peers --- most of the hierarchical primitives are either inheritance-based or containment-based. Containment relationships are defined at runtime and as such are open to being used to support configuration of peer-based arrangements such as a collection of jointly managed components.

This architecture will integrate the management object model based on MIBs with the implementation object model based on CORBA by defining example CORBA-based jargons for each of the major MIB information types.

20.4. Distribution

20.4.1. Distribution Between Peers

The first distribution requirement is that the systems exchanging management data do so as peers. One system shall not be dependent on the other to operate (although it may not have access to as many managed objects). However, hierarchical or centralized arrangements, while not the target configuration, will not be prevented.

Because the peer systems will have their own collateral services, such as managed object directory services and local clocks, integration and distribution of information from such peer collateral services is also necessary.

20.4.2. WAN Topology

As networks expand, they are increasingly forced to interoperate over *Wide Area Network* (WAN) interfaces with peer networks run by peer organizations using their own network management systems. Support for WAN-based distribution across such borders is therefore required.

Another driving force for WAN distribution is the nature of ATM equipment management interfaces (support for which is another Peer-to-Peer system requirement, see above). A WAN channel is often the initial link between organizations using two management systems to deploy ATM equipment, much like a lightweight heaving line is used to pull a heavy mooring hawser across.

The primary effect on the system architecture was ensuring that the CORBA ORB distribution framework properly supports WAN distribution. These issues are covered below, under *Coping with CORBA Deficiencies*.

20.4.3.Distribution of Management Control and Information

The management information distributed shall include status monitoring, event monitoring, issuing control commands and exchanging historical data to provide context. These four functions exemplify both the major application functions (monitoring and control) and the major usage patterns (request/ response, updating large datasets or infobases and asynchronous notification).

The functional and usage characteristics jointly define a jargon, an inter-system transfer mechanism whose characteristics are determined by the information and its usage. For example, management application features such as historical collections should be fully accessible from a number of different management systems. Distribution of large datasets of potentially processed management information is different from distributing raw information of the specific devices.

20.4.4.Standard Distribution Framework

As discussed in the Requirements Document, we consider an object-oriented distribution standard necessary to minimize the number and methods of low-level mechanisms used for distribution. Therefore, we have chosen the Object Management Group's CORBA standard as providing most (although not all) of the necessary low-level distribution capabilities while simultaneously supporting the object-oriented requirement. The CORBA standard was chosen because of its open architecture as well as the availability (and maturity) of its distributed object management and distribution services. It was also chosen because CORBA ORBs are increasingly seen as the distribution mechanism standard of choice within management systems, which should enhance both the long-term extensibility and reduce the long-term cost of management systems.

20.4.5.Coping with CORBA deficiencies

There are a number of difficulties in using CORBA to fulfill all the requirements discussed above:

- I. Lack of CORBA support in management systems
- II. Lack of CORBA V2 IIOP support in management system ORBs
- III. CORBA's reliance on LAN-centric synchronous RPC for transport
- IV. CORBA's lack of full and sufficient specifications for Collateral Services:
 - A. Directory Services that are not LAN-centric
 - B. Time Services that are not implicitly assumed or based on independent local clocks
 - C. Security Services
 - D. Management Services that manage the CORBA ORB and the P2P system extensions without depending on them recursively.

- E. Persistence Services that go beyond writing every operation to disk and which are customizable on a per-object basis

Although many management systems are object-oriented as far as the managed objects are concerned, the implementation of the system itself is not because of the need to distribute management information. If such information is held by system objects, they must be distributed and standard support for distributed objects is a recent technological development. Furthermore, even when the management system is based on an ORB, inter-operation with that ORB may be difficult, due to licensing and version skew problems. The standardization of the low-level protocol for Internet InterORB Operability has happened only recently, and its usage conventions are still not fully worked out beyond the demo stage.

This architecture intends to address inter-system communications issues by selectively exporting and distributing management information, and providing a limited number of standardized, CORBA-based wrappers for non-CORBA-compliant systems (such as HP Openview) where direct access to the CORBA ORB framework is not available. When an ORB is installed but its framework is not available, the use of an auxiliary ORB and standard CORBA APIs should minimize the later effort of porting a jargon to the previously unavailable embedded ORB.

The underlying transport used by CORBA for procedural language mappings like C++, C and Java Remote Method Invocation (RMI) is Remote Procedure Call (RPC). This is a synchronous blocking mechanism, which we find inadequate for many types of management information usage, especially volatile types such as status and unpredictable ones such as events. Therefore, this architecture will address standard request/response usage with jargons directly defined by CORBA IDL members and attributes. We define non-request/response usage or indirect access jargons based on *PASS objects*: each PASS object is a CORBA-based transport object typed according to the management information it carries. PASS objects provide an experimental asynchronous notification service implemented using an equally experimental implementation of C++ templates in the IDL compiler.

Lack of a well-specified object instance directory service (or Implementation Repository) which complements the existing Interface Repository (type and other meta-information) has been a significant drawback of the CORBA V2 standard that has only recently been remedied. The standard still does not directly address the WAN distribution requirement, and CORBA services are of no use out on the WAN if an ORB cannot locate them when asked. However, it does specify a Directory Service interface, which can be integrated with other directory services whose WAN requirement is well understood and supported. This architecture recommends that if the ORB's implementation of the name service is inadequate, the system should supply and utilize another.

Lack of an explicitly specified distributed time service, which allows for coordinated, accurate timestamps from multiple systems (including non-computer-based clocks) is also a significant drawback. Coordinated time is not a critical service for distributed object brokering per se, but is required for managing distributed processing among heterogeneous systems. CORBA implicitly depends on one of the two most widespread time services,

NTP and manual synchronization. Full integration of CORBA and DCE (which has started with the DCE Name Service) would also remedy this lack by providing the DCE Time Service.

Lack of a fully specified distributed Security Service is being addressed by the Object Management Group, but in the interim, the issues must be addressed locally. See the next section, on Security Integration.

Lack of a well-specified distributed management architecture is noticeable mainly in dealing with managing ORBs themselves, particularly when the management services are ORB-dependent and yet must function before the ORB is fully configured and functional. In peer architecture, with potentially many ORBs, this is particularly apparent. These issues are being addressed locally; see the section on System Management, following.

Lack of a persistence or storage interface is most noticeable in the lack of efficient implementations. These issues are being addressed locally; see the section on Storage, just below.

20.5. Security Integration

Whenever resource control flows are distributed across a network, security concerns become more prominent. Such security needs to cover from the lowest physical level up through the application level. First, the integrity of network management functions and information needs to be assured, particularly for control functions. Second, authorization and access control policies should be supported, including privacy support for sensitive information. Third, minimal protection against replay attacks is desirable, although extending it to a comprehensive defense against denial of service attacks is not cost-effective. Finally, derived requirements arising from the need to extend integrated security to WAN environments via common security mechanisms such as firewalls will be considered Management of Security of Management component issues; only hooks or placeholders will be defined or provided. This architecture will define such hooks for all relevant security mechanisms provided by the two chosen information management systems, plus any relevant ones specified by Odyssey Research Associates in their architecture.

20.6. System Management

The system itself will provide a management interface and GUI that will enable the system integration components to be installed and configured. As such integration components will themselves be jointly managed by the peer systems, recursion issues will be dealt with by specifying low-level or minimal subsets for critical services.

20.7. Storage

The system shall support usage-determined storage to complement the distribution methods. Just as the usage patterns drive the choice of jargon, they will also drive decisions about when and where information will be persistently stored. Persistence mechanisms will either conform to the CORBA standard or their usage will be explicitly specified as a distinct, replaceable module and access to that information will conform to MIB/CORBA standards via a jargon.

20.8. Graphic User Interface

The system GUI shall be compatible with the HP Openview Xlib/Motif GUI and Tivoli Systems TME at the X Windows level. It shall also allow Motif-compatible HTML viewers to be installed on the systems, such that displays from the XBind CGI scripts may be displayed. Additional GUI components to support and provide access to integration components will be defined in either Tcl/tk or as HTML pages suitable for viewing via browsers.

21. Functional Component Description

The *Functional Component Description* contains a lower, but still functionally oriented breakdown of the major components of the system. Items in this section are described in relation to how they provide system functionality instead of how they function as discrete applications. This means that the Graphical User Interface (GUI) would be described in terms of the operator functionality it supports, rather than the method it would use to interface with network devices.

Integrating two peer management systems requires a model of how to share management tasks in a peer-to-peer fashion. Management Systems are applications, which hold a local view of one or more managed resources, the actual instance of which may be remote. (See the Management System diagram below.) The management view of a managed resource is termed a managed object and it holds a number of items of information about itself, including its resource type, ID, status, relationships with other managed objects etc. (See Managed Object diagram below). Managed Objects are defined by their class definitions, also known as their *management information base* (MIB). When resources are remotely managed across the network, access to the MIB may be either through an application/agent management protocol such as SNMP or by distributing the Managed Object with a distributed object infrastructure. If a distributed object approach is used, the client M.O. is incorporated into the management system's application and the server M.O. into the managed resource.

Our model for peer sharing of such managed objects is intended to cover 1) the situation in which the two systems have 'joint custody' of some managed resource or 2) they provide 'continuous coverage' as a parallel load sharing arrangement or 3) they have a sequential arrangement in which there is a changing of the guard. In all of these cases, there is a period in which parallel management or 'joint custody' is desirable. The key difference between a peer management system architecture and a hierarchical one is based on the additional requirements generated by the need to coordinate decisions in a joint custody arrangement.

To support joint management we define shared managed objects. A shared object can be any of the managed objects definable in an information system management model. The most familiar of such objects are the real network components or devices (as in an SNMP-accessible Managed Object Instance), but they can also be management-system-defined objects. One such familiar case are the collections of management information intended to be transferable between network management systems, such as an event log or a historical statistical dataset.

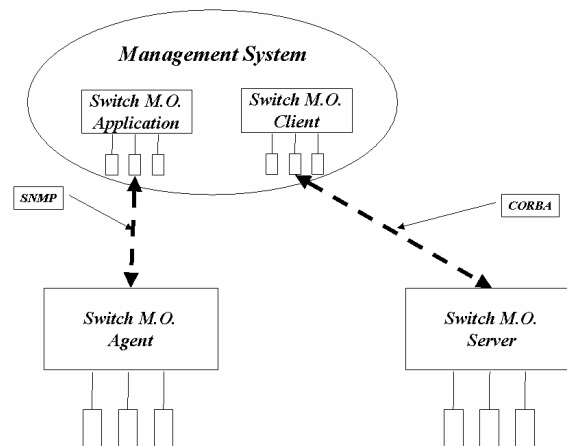


Figure 21-1 A Management System

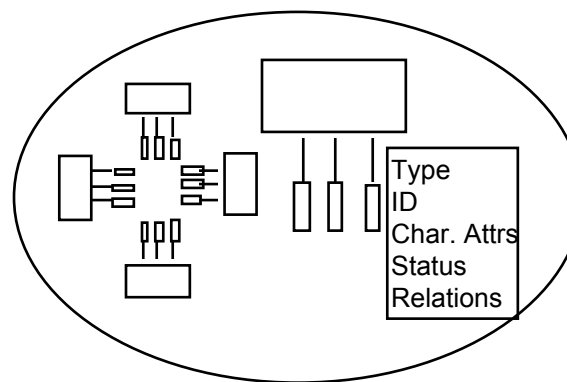


Figure 21-2 Managed Object Structure and Formal MIB Definition

Managing shared objects is based on the basic management functions, monitoring and control. Making such standard managed objects *shared* involves two additional requirements:

1. Distributing monitoring information equally to all responsible peer management systems
2. Synchronizing control responsibility

Monitoring and controlling a shared object can be done in several ways, but they are all based on designated peer management systems exporting attribute values collected locally to a distributed shared object and all other management systems then monitoring that shared object. The simplest form of monitoring and control involves using a simple CORBA distributed object to export the managed object from a designated *local system* (see [the section](#) on Basic Management of Shared Objects, below). All the other peer systems then use their standard monitoring and control processes to monitor and control that distributed object. This approach has its drawbacks however, involving coordinating

control of monitoring policy, which we discuss in more detail below. It also cannot handle asynchronous notification easily, forcing event monitoring into another special case. However, it is the normal approach for sharing control.

The Shared Managed Object figure below shows two management systems sharing such an individual managed object, Switch, which the management system on the left is exporting. In addition they are sharing a collection of historical data, an example of an alternate monitoring approach.

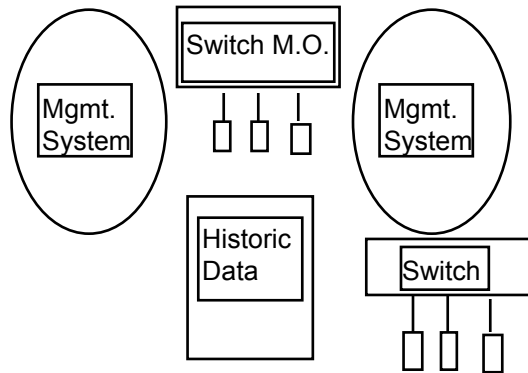


Figure 21-3 Shared Managed Objects

This alternate approach delegates all monitoring responsibility by treating the managed objects to be monitored as a collection. Monitoring is the most common case of managed objects being treated collectively, because both daily operations and statistical analysis require on-going monitoring of groups of managed objects. In those cases, making the resulting management information and operations shared may be more efficiently implemented by making the collection as a whole shared.

Transfer of collective information between peer systems can then be optimized if necessary by transferring the monitoring policies (target objects, poll interval, optional access control, etc.) and their control to the local management system. The collected and partially digested monitoring results are in turn transferred from that collector management system.

This delegation approach is used for collecting the historical data used to manage trends. For example, sharing historical statistics would involve setting up shared statistical application objects which represent the historical datasets and providing an administrative interface that would control when and how the datasets were distributed.

Managing asynchronous notifications or events from shared managed objects is often also managed via this delegation approach. The shared object in this case is a collective event source that may have also filtered or combined events from upstream sources or even generated additional events based on incoming events.

21.1. Basic Management of Shared Objects

The simplest management tasks involve low-level monitoring and control operations and basic shared management is based on a direct jargon that defines MIB access via CORBA IDL.

Applying a GET or SET to a shared object's IDL-specified attribute is simply an extension of applying them to a regular managed object and involves:

1. Initiating the command from within the first NMS or from its GUI.
2. In the case of SET operations, verifying control access is permitted.
3. Requesting the information from or operation on the shared object via the jargon interface. The request is forwarded to the responsible peer system for local action and the result awaited.
4. Invoking the translator wrapper for reformatting the result into the local format if needed.

21.2. Asynchronous Attribute Monitoring

When monitoring a shared managed object's attribute(s) via a series of synchronous polls or GETs consumes too many resources, an alternative shared management interface can be defined. This is often the case with status attributes, which are either so stable or so volatile as to be barely worth GETting even if only one management system is involved, let alone one or more peers. In any case, we define a requirement for asynchronous monitoring in which the local management system is responsible for GETting the attribute value(s) in question and then publishing them via a distributed shared object for peer systems (including possibly itself) to monitor by subscribing to the shared object.

21.3. Shared Historical Context Data

Sharing data such as trend and other historical information is a combination of sharing standardized management data and non-standard application data in a standardized fashion. Sometimes context data are standardized (such as MIB-defined trend data) and can be embodied in a peer management system via a shared standard managed object. Other times, this is not so. In the latter case, we can define a private managed object class that is then shared. The underlying mechanisms for doing this are:

1. Initiating historical data collection and local storage from the component sources from within the transmitting NMS or from its GUI.
2. Translating or reformatting the historical data if needed.
3. Distributing the collected data via a jargon, possibly integrating them with previously distributed data in a distributed object that serves as the basis for the 'shared object' view in each participating management system.
4. Initiating the receiving of distributed data from within the receiving NMS or from its GUI.
5. Translating or reformatting the historical data if needed.

21.4. Shared Object Control Operations Coordination

As mentioned earlier, the second of the two major distinguishing features of peer management architectures is coordination of *control operations*. Control operations are those involving writes, either as actions or sets. In order for a target shared managed

object to present a consistent view of its state to each of its responsible peer managers, control access must be both properly *authorized* and *sequenced*. Authorization of operations actually consists of two functions, *authentication* and *authorization*. In sharing control responsibility, one peer management system delegates authority to, and defines a trust relationship with, another. Such a trust relationship is built on a series of authentication and authorization handshakes starting with the human operator launching the initial management application GUI, which in turn initiates programmatic actions which are distributed by the jargons, which in turn initiates actions by the receiving peer system. Occasionally, the relationship is extended all the way to the ultimate target managed object.

The initial authentication in this version of the architecture is based on a simple user ID and password mechanism as supplied by the two information management systems, OpenView and Tivoli. Such authentication mechanisms will be secure only to the extent that their vendors have provided mechanisms. Kerberos/DES in Tivoli may provide some significant protection, for example. However, both OpenView and Tivoli's principals are based on the UNIX login IDs, and in the absence of Kerberos, even more dependent on a secure underlying operating system. This architecture assumes that, when necessary, secure UNIX kernel and inline network encryption mechanisms, such as FastTrack, will be added. Authentication following the initial operator validation is based first on CORBA mechanisms, followed by the server supplying the receiving peer system with an appropriate user identity.

Authorization in this version of the architecture is based on a simple, platform-supplied scheme in which per-object access control sets store the authorized users' IDs.

Sequencing of control requests is done on a first come, first served basis as received by the object's server.

21.5. Collective Event Sources

The simplest event management involves defining or configuring what types and sources of events will be redirected or copied to the peer system. The underlying mechanisms for doing this are:

1. Initiating the collection from the component sources from within the transmitting NMS or from its GUI.
2. Translating the event data if needed.
3. Transferring the event via a PASS jargon to the second NMS.
4. Initiating the collective source and monitoring of the jargon from within the receiving NMS or from its GUI.
5. Invoking the translator wrapper for reformatting, if needed.

21.6. Competing Multiple Sources

There is one case in which collecting management information from multiple sources is undesirable: when the sources compete with each other to supply the same information about the same shared managed object. This can happen when the information

management system both collects its own information and imports it via a jargon or uses two types of jargons to get the same data. There are two approaches offered to deal with this situation: overwrite and parallel objects. In the overwrite situation, the two sources are merged and whichever one writes last determines the current value. This approach can be useful for objects whose initial or complete set of values is updated via one synchronous source and then critical values are updated asynchronously, or when switching from one source to the other with some overlap.

The other approach is to define parallel objects or variant attributes on the platform, with one object or attribute per source. This is useful when the multiple sources are truly competing with each other and disagreements on the value need to be recognized, not overwritten.

Generally objects (data structures) are used for complex information that must be read or written in a single transaction and attributes are used for information that while related to the other fields in the parent object can be read and written independently. The definition is done within the information management system, using its native mechanisms. For example, in HP Openview an `OV_Field` may be used to hold an attribute value imported from a jargon.

22. System Interface Description

The *System Interface Description* describes the relationship of the various system component types described in the System Component Description section. The intent of this section is to describe how the system components interact to allow the functional components to support the functionality described in the Functional Component section.

There are three major types of interactions among components. First are the configuration and translation/formatting interactions between management system components and the jargon components. Second is the distribution of the categorized-by-jargon management information between the client and server parts of jargon components. Third is the management of the jargons that make an object shared. The following sections describe the system interfaces that allow peer management system components to utilize jargon-distributed information and how such translation and distribution mechanisms are configured.

The information in the MIB and applications has a range of distribution requirements. Some information is stable and is required to persist across system re-boots, other information is extremely volatile. Some information comes in large packages, other information is compact and does not require much bandwidth to transmit. Some information transfers are initiated by the management system, others by the managed object.

22.1. Jargon-based Distribution

The interface between a management system and the jargons is a significant portion of the system architecture. There are two main approaches, the *direct* approach that utilizes CORBA to define jargon system interfaces directly and the *indirect* approach used when the direct approach is unavailable or inappropriate.

A direct interface is one specified in IDL; it is straightforward and is covered by the CORBA and Corbus documentation referenced. Often such managed objects are generated by MIB-to-IDL compilers such as SNMPIDL. However, the easiest way to integrate managed objects so specified in a management system is to provide a CORBA-compliant ORB framework in which the object and its interface can reside. Unfortunately, such management systems are rare (although becoming less so) and the ones that exist put a premium price on installing the managed object into the CORBA ORB framework.

Even if access to a CORBA framework is available, direct interfaces have drawbacks that make them inappropriate for distributing some types of management information. In particular, the heavy reliance of CORBA on an underlying synchronous and quite heavy RPC transport mechanism makes it both expensive and cumbersome to use for transient or stream-like information.

However, this drawback can be hidden by indirect use of a CORBA based PASS object to provide a distribution channel that is almost as efficient as an asynchronous message-based channel while it is as correct and reliable as the synchronous channel on which it is based. As is often done in the communications world, the cost of the underlying synchronous channel is often shared by a number of multiplexed asynchronous users.

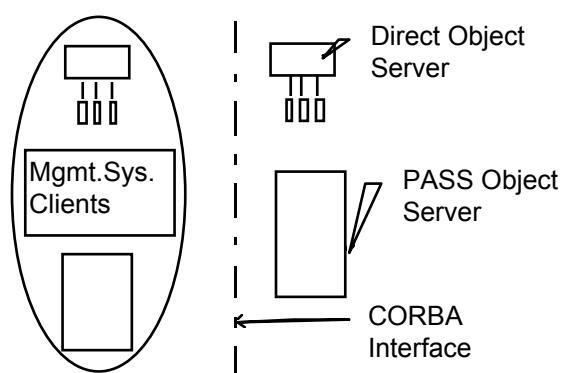


Figure 22-1 Jargon-based Shared Managed Objects

Using a PASS object to distribute information requires additional components to read, write and multiplex to the channel, these actually provide the API for the peer management systems. Management System-specific translation and format processing can be integrated into the reader/writer components, as can distribution policies such as attribute value edge detection or delta detection.

For example, in the status jargon known the Management Information Status Tracking (MIST), whose core is the STATUS_PASS CORBA object, the server process implements the policy of how information is distributed to the clients. Once a subscriber has received a complete copy of all the elements of the object, only new or different elements will be sent. Enforcing this restriction is not placed on the publishing clients; the server will determine if the received values are duplicates of what it has already stored.

Jargons based on PASS objects will allow external clients to perform the following methods on the application meta-objects:

1. Get notified of new object values

2. Get sent new object values automatically
3. Define new objects
4. Define new object values
5. Extract particular object values
6. Delete old object values

The ability to perform these MIST functions will be controlled by the STATUS_PASS, PASSWRITER and PASSREADER methods. For example, the current *STATUS_PASSWRITER* interface has the ability to publish the operational status of a set of managed objects by updating the STATUS_PASS object.

The reason for this generic PASS object approach is to allow datatype- and management system-specific components to exist independently from the specifics of the distribution and subscription mechanisms. This means that any management information datatype to be distributed can be mapped onto any appropriate distribution method without affecting the external portions of the jargon definition.

In addition to the fundamental work of distributing management information through various kinds of jargons, the peer management systems must also be able to configure, monitor and otherwise manage these extensions that permit managed objects to be shared. Managing jargons and other meta-management tasks have an additional constraint: they must pull themselves up by the bootstraps and not rely on jargon-based mechanisms themselves for configuration information such as timestamps, authentication, existence and location information unless that particular collateral services sub-system is already up. This constraint therefore requires that a *startup sequence* or *dependency* graph be specified for each site's configuration in which the source of the meta-management information at each stage of startup and shutdown is specified. Such graphs may be used only by manual configuration systems or may be automated such that the SMO Share operation (see below, SMO Operations) control flow is based on them.

22.1.1. Direct IDL Jargons

Direct IDL Jargons actually have two functional interfaces defined by the formal IDL interface. The first uses a combination of standard Create, List and Delete on a management system-specific factory to instantiate, browse and remove managed objects shared by management systems of one type with management systems of the same or another type. The second uses the CORBA standard *Attribute* not only to define the datatypes of management information but also the read and write operations --- the GET and SET methods --- in IDL terms. This is the IDL often generated by a MIB-to-IDL compiler tool. Therefore all direct IDL jargons have the form:

```
interface directSharedManagedObject
{
    // This interface IDL is often generated by a MIB to IDL compiler, then included in the
    overall IDL
    // Datatypes
    //     These are SMO class specific
    // Exceptions
```

```

//      These are SMO class specific
// Operations
//      Since the GET and SET operations are implicit in the attribute specification, this
section is
//      usually empty
// Attributes
//      List all shared attributes for this SMO class here. This is the workhorse section of
the IDL
    attribute MIBVarType MIBVarName;
};

interface directSharedManagedObjectFactory
{
    // This interface may have to be hand-coded and matched to the SMO interface
    // Exceptions
        exception SMO_invalidArg {long argTypeCode, string message};
        exception SMO_exNoSuchObj {string objStringRef};
    // Operations
    //      These operations are factory standards
    // create Shared Managed Object
        SharedObject Create(in sharedObjArg1, in sharedObjArg2,...)
            raises ( SMO_invalidArg,...) ;
    // destroy Shared Managed Object
        void Destroy();
    // list Shared Managed Objects
        sequence<SharedObject> ListAllObjects();
    // Attributes
    // Factory attributes tend to be implementation specific.
};

```

22.1.2.PASS-based Jargons

The PASS is a container class that serves as a kind of intelligent buffer between producers (writers) and consumers (readers) for any user-supplied type of data. Multiple writers may populate a PASS object, and the data they provide may fan out to multiple readers. Not all data supplied by writers will be forwarded to readers; the details of the filtering done by PASS are described in the PASSREADER and PASSWRITER sections for each PASS-based jargon.

22.1.2.1.PASS IDLs

```

// File:   pass.idl
// Contents: CORBA IDL specification for Pass
// System:  P2P development.
// Created: 14-Jan-1997
// Author:
// Remarks:

```

```
// Generated by ldi on Tue Jul 11 16:40:42 1995.
// $Header: /nfs/morpheus/u3/p2p/rcs/doc/techreports/sa1-10.rtf,v 1.2 1997/10/16 21:05:08
lbob Exp $
// COPYRIGHT 1997 BBN Systems and Technologies
// A division of Bolt, Beranek and Newman, Inc.
// All Rights Reserved.
//
// 10 Moulton Street
// Cambridge, Ma. 02138
// 617-873-3000
```

22.1.2.1.1.PASSREADER IDL

```
interface PASSREADER
{
    void Destroy();
    // Display PASSREADER object
    void Display(
        out string pReaderName,
        out unsigned long task,
        out string RegisteredOnPass
    );
}; // interface PASSREADER
```

22.1.2.1.2.PASSWRITER IDL

```
interface PASSWRITER
{
    void Destroy();
    // Display PASSWRITER object
    void Display(
        out string pReaderName,
        out string RegisteredOnPass
    );
}; // interface PASSWRITER
```

22.1.2.1.3.PASS IDL

```
interface PASS
{
    //
    // Exceptions

    exception PASSWRITER_INV {} ;
    const string PASSWRITER_INV_msg
        = "PASSWRITER object invalid or does not exist.";
    exception PASSWRITER_MAX_SIZE {
```

```

        unsigned short MaxSize ;
    } ;
const string PASSWRITER_MAX_SIZE_msg
    = "PASS object contains maximum number of PASSWRITERS.";
exception PASSWRITER_NOT_REGISTERED {} ;
const string PASSWRITER_NOT_REGISTERED_msg
    = "PASSWRITER has not been added to this PASS object.";
exception PASSWRITER_ALREADY_REGISTERED {} ;
const string PASSWRITER_ALREADY_REGISTERED_msg
    = "PASSWRITER has already been added to another PASS object.";
exception PASSREADER_INV {} ;
const string PASSREADER_INV_msg
    = "PASSREADER object invalid or does not exist.";
exception PASSREADER_MAX_SIZE {
    unsigned short MaxSize ;
} ;
const string PASSREADER_MAX_SIZE_msg
    = "PASS object contains maximum number of PASSREADERS.";
exception PASSREADER_NOT_REGISTERED {} ;
const string PASSREADER_NOT_REGISTERED_msg
    = "PASSREADER has not been added to this PASS object.";
exception PASSREADER_ALREADY_REGISTERED {} ;
const string PASSREADER_ALREADY_REGISTERED_msg
    = "PASSREADER has already been added to another PASS object.";
exception PASSREADER_BUSY {} ;
const string PASSREADER_BUSY_msg
    = "PASSREADER is already blocked waiting for a PAYLOAD.";
exception PAYLOAD_INV {} ;
const string PAYLOAD_INV_msg
    = "PAYLOAD object invalid or does not exist.";
exception PAYLOAD_MAX_SIZE {
    unsigned short MaxSize ;
} ;
const string PAYLOAD_MAX_SIZE_msg
    = "PASS object contains maximum number of entries.";

//
// PASS Operations
//

// Unconditionally destroy PASS object
void Destroy();

// "Display" a PASS object. This returns information that is
// useful for sanity checking/debugging. This operation does not
// change the state of the object.

```

```

void Display(
    out string pPassName,          // name of this PASS
    out string pPassType,          // type of this PASS
    out unsigned short pMaxTasks,   // # of tasks
    out unsigned short pMaxRecs,    //
    out unsigned short pMaxWriters, // # of writers that can register
    out unsigned short pMaxReaders, // # of readers that can register
    out unsigned short pRecCount,   //
    out sequence<PASSWRITER> pWriterList, // list of current writers
    out sequence<PASSREADER> pReaderList // list of current readers
);

// Add a new PASSWRITER to a PASS object by name
PASSWRITER AddWriter(in PASSWRITER Writer,
    in string WriterName)
    raises(PASSWRITER_MAX_SIZE);
// Remove a PASSWRITER from a PASS object
void RemoveWriter(in PASSWRITER Writer)
    raises(PASSWRITER_INV);
// Add a new PASSREADER to a PASS object by name
PASSREADER AddReader(in PASSREADER Reader,
    in string ReaderName)
    raises(PASSREADER_MAX_SIZE);
// Remove a PASSREADER from a PASS object
void RemoveReader(in PASSREADER Reader)
    raises(PASSREADER_INV);

} ; // interface PASS

```

22.1.2.1.4.PASSMGR IDL

in file *pass.idl*:

```

interface PASSMGR
{
    //
    // Exceptions
    //

    exception PASS_EXISTS {};
    const string PASS_EXISTS_msg
        = "PASS object already exists.";
    exception PASS_INV {};
    const string PASS_INV_msg
        = "PASS object invalid or does not exist.";
}

```



```

        exception PASS_TYPE_INV {} ;
        const string PASS_TYPE_INV_msg
            = "PASS type invalid.";

//
// Operations
//

// Create a PASS object
    PASS Create(in string PassName,
                in string PassType,
                in unsigned short MaxRecs,
                in unsigned short MaxWriters,
                in unsigned short MaxReaders)
        raises(PASS_EXISTS, PASS_INV);

// Lookup a PASS object given its name and type.
    PASS Lookup(in string PassName, in string PassType)
        raises(PASS_INV);
// PassEntry contains the name and type of a PASS object.
// It is used in the parameter list of the List operation.
    struct PassEntry {
        string name;
        string type;
    };

// List returns the names and types of all PASS objects
// known to this PASSMGR.
    void List (
        out sequence<PassEntry> pPassList
    );
}; // interface PASSMGR

// local types start here.
// do an #include "type.idl" to pull them in
// They in turn will pull in the PASS Template IDL

```

22.1.2.1.5.PASS Template IDL

```

interface PAYLOAD
{
    PAYLOAD_STRUCT
    void Display(out PAYLOAD_REP ppRec);
    void Destroy();
};

```

```

interface PAYLOAD_PASS : PASS {
    PAYLOAD_STRUCT
    // Add/update PAYLOAD to a PASS_Status object
    void AddUpdateRec(in PASSWRITER Writer, in PAYLOAD_REP pRec)
        raises(PAYLOAD_INV, PAYLOAD_MAX_SIZE);
    // Remove an existing PAYLOAD from a PASS object
    void RemoveRec(in PAYLOAD_REP ppRec)
        raises(PAYLOAD_INV);
    // Lookup a PAYLOAD in a PASS object
    void LookupRec(inout PAYLOAD_REP ppRec)
        raises(PAYLOAD_INV);
    // Lookup next PAYLOAD object
    void LookupNextRec(in PASSREADER Reader, out PAYLOAD_REP ppRec)
        raises(PAYLOAD_INV);
};
#undef PAYLOAD
#undef PAYLOAD_STRUCT
#undef PAYLOAD_REP
#undef PAYLOAD_PASS

```

22.1.2.2. Object Interactions in PASS

The following diagram shows a typical pattern of messages and object creation that might be seen in a system using PASS objects to transfer Status. The commentary explains the objects involved, the effects of the messages, and what is going on behind the scenes.

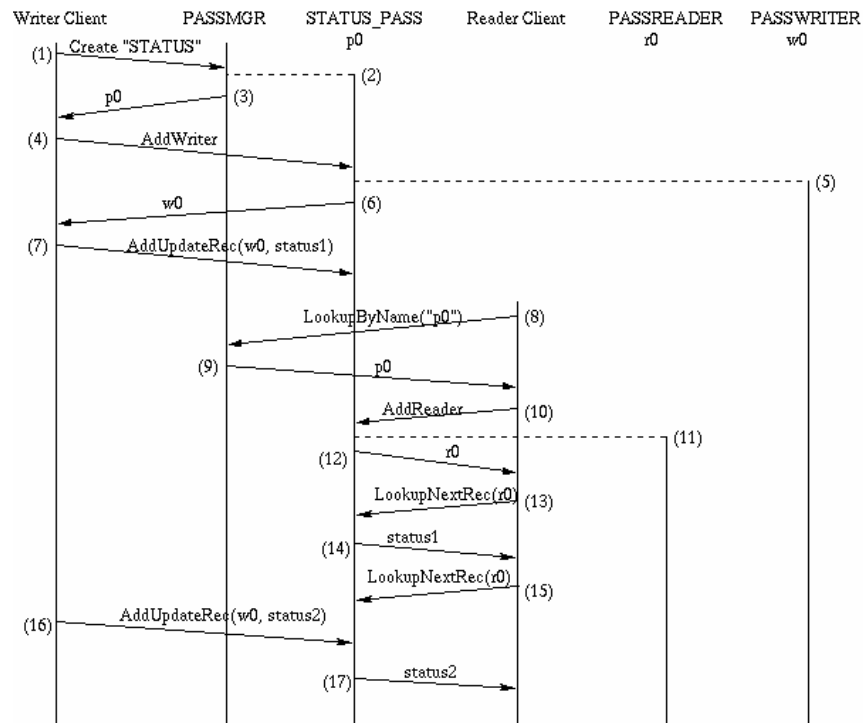


Figure 22-2 Example PASS Jargon Usage

1. The writer locates a preexisting PASSMGR object, either by retrieving its object reference from the PASSMGR_OBJECT environment variable or via the Corbus function `bind_to_any_in_clust()`, and sends it a create message specifying it needs a PASS object through which STATUS objects can be passed.
2. The PASSMGR object is a factory and lookup service for PASS objects. Here it fulfills its factory duty by creating a STATUS_PASS object in response to the Create request in (1). Note that pure PASS objects are never created; only subclasses of PASS that are specialized to carry a particular data type can be created. In C++ parlance, PASS is an abstract base class.
3. PASSMGR returns the object reference of the created STATUS_PASS object, p0.
4. Before the writer client can begin pumping data into the STATUS_PASS object p0, it must register itself as a writer by invoking the AddWriter operation on p0.
5. In response to the AddWriter, p0 creates the PASSWRITER object and returns its object reference, w0, in step (6). Alternatively, the writer could have passed in a previously created PASSWRITER object. Internally, p0 remembers that w0 is now registered for writing on p0.
6. The writer now presents the first piece of data, status1, to p0 with the AddUpdateRec operation. The writer identifies itself to p0 by supplying the PASSWRITER object w0 from step (6). Currently, PASS objects make little use of the writer identity, but it could be used in the future to implement various policies, e.g., prioritization of the data to be distributed to readers based on who wrote it.
7. Attention now shifts to the reader side. A reader client wants to receive data from a PASS object named "p0". It finds a PASSMGR object using methods similar to those described in step (1), and invokes the LookupByName operation on the PASSMGR object, which returns the object reference p0 in step (9). PASSMGR objects also support an operation that lists all available PASS objects by name. These operations comprise the "lookup service" mentioned above.
8. Exactly paralleling the writer registration process, the reader client does an AddReader on p0, and p0 creates the PASSREADER r0 (11) and returns it (12).
9. The reader client asks for the first piece of data from p0, and p0 returns status1 (14) that was written in step (7).
10. The reader client asks for the next piece of data from p0, but since there is none, it does not immediately receive a reply. Internally, p0 suspends the thread that is servicing this operation.
11. The writer sends another piece of status information, status2.
12. p0 awakens the thread suspended in (15) so that it can return status2 to the reader.

22.1.3.Jargon Management and Shared Managed Objects

To manage a Shared Managed Object consists primarily of identifying *which* underlying Managed Objects will be shared, with *whom* they will be shared and enumerating *the data and operations* that will be shared using jargons. To do this we define the following six classes:

1. **Peer Manager Set:** a set of one or more *Peer Managers*, comprising the known, permitted and available peer managers. There are at least two instances of this class, one of all possible managers, known as the *Peer Manager Registry*, the other associated with the formal SMO definition, known as the *CustodianList*. Offers the normal set-type operations such as *add*, *remove* and *browse*.
2. **Peer Manager:** a managed object representing a remote management system. It is defined by the Peer Manager Attributes, below, and offers the normal object operations of Create, Destroy and Display.
3. **Shared Managed Object Set:** a set of zero or more *SMOs* comprising the managed objects shared by this management system with the remote Peer Manager. Called *SMOList* in the jargon definition sections below. Offers the normal set-type operations such as *add*, *remove* and *browse*.
4. **Shared Managed Object:** a managed object whose information is imported or exported for sharing. It is defined as a collection of *jargons* connecting a local *managed object* with the remote *peer managers* in the *Custodial Peer Manager Set*.
5. **Jargon Set:** a set of one or more *Jargons* comprising the exported and imported information belonging to a local managed object. The normal set-type operations of add and remove are subsumed by the SMO operations *share* and *unshare*. Called *localJargonList* in the formal SMO definitions below.
6. **Jargon:** a mechanism for importing or exporting a particular type or class of management information. There are two families of jargons, *direct* or synchronous request/response based and *indirect* or Piecewise Asynchronous Service (PASS) based.

22.1.3.1. Peer Manager Attributes

1. **remPeerSysName:** The host, system or node name of the remote peer management system: a human readable string that can be mapped into the network address used by management by a directory service. In this version of the architecture, a DNS name.
2. **remPeerIPAddr:** The network address used of the remote peer system, used when the directory service is not available or accurate. In this version of the architecture, an IP v6 address.
3. **remPOC:** The Point of Contact information for manual administration coordination, this attribute set includes the person's name, organization, address type (email, phone, postal, etc.) and address.
4. **remPlatformType:** The type name of the remote peer management platform. In this version of the architecture, an enumeration, in this version of the architecture, of HPOV or TME, for HP OpenView and Tivoli TME respectively.

22.1.3.2. Shared Managed Object Attributes

localMOID: The Managed Object ID: the name or stringified object identifier of the local managed object to be shared. Used as entry in the *SMOList*. Must include the *sysName* and any subsystem identification necessary to fully distinguish the managed resource for the local information management system.

localJargonList: A list of the supported jargons, specified as a sequence of items from the jargon enumeration. In this version of the architecture, the jargon enumeration consists of Status, Trend Bulk, Trend Subscription, Snapshot, Control and Event. Selected jargons are activated by the Share operation.

localShareType: The direction of the jargon information and control flow; whether the local management system is exporting or importing the managed object information.

CustodianList: the set of names or stringified object identifiers of the remote Peer Managers which share the management responsibility for this SMO.

22.1.3.3.SMO Operations

Share extends a Managed Object or Collection by setting the SMO extension attributes, creates a SMOList entry and starts any jargons specified, including any necessary wrappers.

Unshare stops the wrappers (which may also discontinue the jargon server) and removes the SMOList entry and SMO attributes.

Browse lists all the shared managed objects currently exported by the jargon(s) between platforms of the local type and those of the remote type.

22.2. Asynchronous Attribute Monitoring Interface

The most critical of the attributes that require asynchronous monitoring are status attributes. For this reason, the first asynchronous attribute jargon will distribute status information. Status information about IP managed objects can be determined either by polling SNMP agents for status information (such as Uptime) or by querying the standard status interfaces on the information management system (such as the OpenView managed node status attribute). This information then will be forwarded to a PASS-based status jargon known as the *STATUS_PASS* to be distributed to the peer systems. Each *STATUS_PASS* holds information about a single managed object in a status record datatype used by the template process to refine the PASS into a *STATUS_PASS*.

The resulting *STATUS_PASS* records contain:

1. Network Address
2. Operational Status
3. HeartBeat information

The *Network Address* attribute is an address used to identify a single interface on a managed system. In the case of a network host, this may be the only interface and devices like routers or switches can support multiple interfaces.

The *Operational Status* attribute holds a single value whose type is determined by the status translation tables for the peer management systems in question, but which most often is derived from the enumeration defined for Status by the management platform or occasionally by the MIB.

The HeartBeat attribute is a jargon management field. It is discussed in the management services section.

Once a management system has subscribed to status information, it will continue to receive status items (as *STATUS_PASS records* until the subscription connection is terminated.

Supplying status from non-SNMP sources will require a suitable *Status PASSWRITER* function which can take in status from the non-SNMP source on a specified regular basis and provide a standard translation if necessary. It will also require a suitable *Status PASSREADER* function which will provide a library of the inverse read functions.

22.2.1.Status_PASS API

The Status_PASS API consists of two operations and a distribution or update policy. The operations include a read operation, which posts a read request via the PASSREADER library, and a write operation, which writes via the PASSWRITER. Distribution policy for status information is only to write a new status record if the incoming status value is new or different. Status information only is given to readers if they are newly created (and hence have not seen any of the data yet) or if the reader is already up but there is a newly written status value. Furthermore, the status data may be declared persistent; if the server crashes and restarts, subsequently registered PASSReader will get all of the current data (per the previous rule).

22.2.2.Status PASSREADER

The status PASSREADER is a library object that embodies the read distribution policy for Status information.

22.2.3.Status PASSWRITER

The status PASSWRITER is also a library object that embodies the write distribution policy for Status information.

22.2.4.STATUS_PASS IDLs

Using the standard CORBA IDL mechanism, `#include "status.idl"`, add the following status refinements as stored in the file *status.idl*

```
#define PAYLOAD_STRUCT
struct STATUS_REP {
string RecName;
string RecValue;
string HeartBeat;
};
```

```
#define PAYLOAD_REP          STATUS_REP
#define PAYLOAD              STATUS
#define PAYLOAD_PASS         STATUS_PASS
// The following pass-template IDL is described in the section Pass Template IDL
#include "pass-tmpl.idl"
```

22.3. Historical Data Collection and Distribution

The source of all trend and other historical context data is a periodic collector of some standard attribute values for one or more attributes. Making those historical datasets available to a peer manager can take several forms.

The basic model of stable historical dataset transfer populates its Trend distributed object on Create or Read. The server takes a snapshot of (for example) the HP Openview historical database files, converts them to the IDL-defined wire format and ships them back to the client.

It is also possible to declare the Trend object persistent and use a storage/retrieval-optimized persistence API to keep a stable copy within the distributed system.

When the historical dataset is volatile, for example when it is still in the process of being created at the source peer management system but the other systems wish immediate access, another Trend jargon is necessary, the *Trend Subscription* jargon. All of the Trend variants of historical context transfer include implementations of the following interface methods. Note, Items c) through e) may be bundled into a single fully distinguished identifier such as an OID.

1. An operation to read a chunk of history from the Trend object. Input arguments include:
 - a) the time collection started (the anchor time)
 - b) the time interval covered
 - c) the source managed system
 - d) the source managed component class ID on that node (if any)
 - e) the source component instance ID on that node (if any)
 - f) the MIB variable ID
 - g) the MIB variable value
 - h) resampling interval required Resampling allows the data to be treated as if collected at regular intervals even if they were not. A zero value means no resampling

It will return a sequence of structures or objects that contain:

- a) the time collection started (the anchor time)
 - b) the time interval covered
 - c) the source managed system
 - d) the source managed component class ID on that node (if any)
 - e) the source component instance ID on that node (if any)
 - f) the MIB variable ID
 - g) the MIB variable value
2. An operation to write a chunk of history to, or create, the Trend object using the same format as the read for input arguments. In some Trend jargons this operations will be implicit in either the Create or Read operation, in which case the Trend server must be collocated with the source of the Trend data.
3. An operation to list/browse the available Trend data. The user will be able to get a list of Trend data, discover what hosts and SNMP variables are recorded in them, and for what time periods.
4. An operation to browse the available Trend objects that hold already fetched Trend data.

The ‘chunks’ of history or TrendTables are optimized for transfer by casting them into one of three forms:

1. by time
2. by variable
3. by device

In all cases, the data is a sequence of samples, and one sample is characterized by the 5-tuple (start time, end time, device, variable, value). However, sending all five fields for every sample would be wasteful, since there will usually be much redundancy. For example, there will probably only be a few devices and variables in a table compared to the total number of samples. So, we define three different table formats to factor out different pieces of the redundant information. This crude form of compression is useful since the trend data can be very large (megabytes). It is up to the client to decide which format to use.

22.3.1.TrendTable by Time

A trend table that is organized by time will factor out the start time from the 5-tuple, so the returned data will be a sequence of:

1. start time
2. sequence of (end time, device, variable, value)

This means that all the samples that were taken at the same time appear together.

22.3.2.TrendTable by Device

A trend table that is organized by device will factor out the device from the 5-tuple, so the returned data will be a sequence of:

1. device
2. sequence of (start time, end time, variable, value)

This means that all the samples that were taken for the same device appear together.

22.3.3. TrendTable by Variable

A trend table that is organized by variable will factor out the variable from the 5-tuple, so the returned data will be a sequence of:

1. variable
2. sequence of (start time, end time, device, value)

This means that all the samples that were taken for the same variable appear together.

22.3.4.Common Trend Datatypes IDL

```
// File:  nmtypes.idl
// Contents: idl include for common types used in network management
// System:  p2p development.
// Created: 15-Jul-1997
// Author:  David P. Wiggins

// Remarks: this file is intended to be included by other idl files

// $Header: /nfs/morpheus/u3/p2p/rcs/doc/techreports/sa1-10.rtf,v 1.2 1997/10/16 21:05:08
lbob Exp $

// COPYRIGHT 1997 BBN Systems and Technologies
// 10 Moulton Street  Cambridge, Ma. 02138  617-873-3000

// Identifier for a device -- just an IP address.
typedef sequence<octet> DeviceID;

// Time in seconds since Jan 1 1970.
typedef unsigned long TimeStamp;

// Difference between two TimeStamps in seconds.
typedef unsigned long TimeCovered;

// Part or all of the MIB object ID (OID).
typedef sequence<unsigned long> MIBVariable;

// MIB Variable's value. Assume everything can be represented by a double.
typedef double MIBValue;
```

22.3.5.Trend Bulk IDL

```
// File:  trend.idl
// Contents: trend bulk jargon interface
// System:  p2p development.
// Created: 17-July-1997
// Author:  David P. Wiggins

// $Header: /nfs/morpheus/u3/p2p/rcs/doc/techreports/sa1-10.rtf,v 1.2 1997/10/16 21:05:08
lbob Exp $

// COPYRIGHT 1997 BBN Systems and Technologies
// 10 Moulton Street  Cambridge, Ma. 02138  617-873-3000
```

```

#include "nmtypes.idl"

struct DeviceVariable {
    DeviceID    device; // IP addr
    MIBVariable variable; // SNMP OID
};

typedef sequence<DeviceVariable> seqDeviceVariable;

enum TrendTableType {
    TrendTableByTime,
    TrendTableByVariable,
    TrendTableByDevice
};

// Parent class of all trend tables.
interface TRENDTABLE {

    // Display: return information about this trend table.
    // Added for debugging; it should either be removed or elaborated
    // to also return the devices and variables in the trend table.
    void Display(
        out TimeStamp    startTime,
        out TimeStamp    endTime
    );

    // Destroy: delete this trend table
    void Destroy();
};

// table organized primarily around the sample time
interface TRENDTABLE_BYTIME : TRENDTABLE {

    // info returned for each time value
    struct DeviceVariableValue {
        DeviceID    device;
        MIBVariable variable;
        MIBValue    value;
        TimeCovered timeSpan;
    };

    struct Time_DeviceVariableValue {
        TimeStamp anchorTime;
        sequence<DeviceVariableValue> dvv;
    };
};

```

```

// GetTable: return trend data  for given the device/variable
// pairs that falls within the starting and ending time period.
sequence<Time_DeviceVariableValue> GetTable(
    in seqDeviceVariable sdv,
    in TimeStamp startTime,
    in TimeStamp endTime
);
};

// table organized primarily around the variables
interface TRENDTABLE_BYVARIABLE : TRENDTABLE {

    // info returned for each variable
    struct DeviceTimeValue {
        DeviceID    device;
        TimeStamp   anchorTime;
        MIBValue    value;
        TimeCovered timeSpan;
    };

    struct Variable_DeviceTimeValue {
        MIBVariable variable;
        sequence<DeviceTimeValue> dtv;
    };

    // GetTable: return trend data  for given the device/variable
    // pairs that falls within the starting and ending time period.
    sequence<Variable_DeviceTimeValue> GetTable(
        in seqDeviceVariable sdv,
        in TimeStamp startTime,
        in TimeStamp endTime
    );
};

// table organized primarily around the devices
interface TRENDTABLE_BYDEVICE : TRENDTABLE {

    // info returned for each device
    struct VariableTimeValue {
        MIBVariable variable;
        TimeStamp   anchorTime;
        MIBValue    value;
        TimeCovered timeSpan;
    };

```

```

struct Device_VariableTimeValue {
    DeviceID device;
    sequence<VariableTimeValue> vtv;
};

// GetTable: return trend data for given the device/variable
// pairs that falls within the starting and ending time period.
sequence<Device_VariableTimeValue> GetTable(
    in seqDeviceVariable sdv,
    in TimeStamp startTime,
    in TimeStamp endTime
);
};

```

22.3.6.Trend Bulk Factory IDL

From the file trend.idl:

```

interface TRENDFACTORY
{
    // Added these to get around Corbus limitations.
    typedef MIBVariable TFMIBVariable;
    typedef DeviceID TFDeviceID;

    // CreateTable: return a table object of the given type that has values
    // for the given device/variable pairs over the given time interval.
    TRENDTABLE CreateTable (
        in seqDeviceVariable sdv,
        in TimeStamp startTime,
        in TimeStamp endTime,
        in TrendTableType tableType
    );

    // GetSummary: return all possible variables and devices for which
    // trend data is available, as well as the minimum and maximum
    // timestamp of all the data. This operation can be used to help
    // decide what parameters to give to CreateTable.
    void GetSummary(
        out sequence<TFMIBVariable> vars,
        out sequence<TFDeviceID> devices,
        out TimeStamp minTime,
        out TimeStamp maxTime
    );
};

```

22.3.7.Trend Subscription API

The Trend_Subscription or Trend_PASS API consists of two operations and a distribution or update policy. The operations include a read operation that posts a read request via the PASSREADER library and a write operation that writes via the PASSWRITER.

Distribution policy for individual TrendTable records is only to write a new TrendTable record if the incoming TrendTable record value is new or different. TrendTable record information is only given to readers if they are newly created (and hence have not seen any of the data yet) or if the reader is already up but there is a newly written TrendTable record value. Furthermore, the TrendTable records may be declared persistent; if the server crashes and restarts, subsequently registered PASSREADERS will get all of the current records (per the previous rule).

22.3.7.1.Trend Subscription PASSREADER

The Trend_Subscription PASSREADER is a library object that embodies the read distribution policy for TrendTable record information-- it reads any changed TrendTable records.

22.3.7.2.Trend Subscription PASSWRITER

The Trend_Subscription PASSWRITER is also a library object that embodies the write distribution policy for TrendTable record information--it detects any changed TrendTable records and writes them into the TrendPASS.

22.3.7.3.TREND_PASS IDL

In the standard PASS IDL file pass.idl, and using the standard CORBA IDL mechanism, #include "Trend_PASS.idl", add the following Trend_Subscription refinements as stored in the file *Trend_PASS.idl*

```
#define PAYLOAD_STRUCT
struct TREND_SUBSCRIPTION_REP {
    string DevName;
    string VarName;
    string VarValue ;
    TimeStamp startTime;
    TimeStamp endTime;
    string HeartBeat;
};

#define PAYLOAD_REP          TREND_SUBSCRIPTION_REP
#define PAYLOAD              TREND_SUBSCRIPTION
#define PAYLOAD_PASS        TREND_SUBSCRIPTION_PASS
// The following pass-template IDL is described in the section PASS Template IDL
#include "pass-tmpl.idl"
```

22.4. Attribute Snapshots

Getting the latest value of a shared managed object's attribute is often useless without context, but acquiring the current value and acquiring the context are modeled here as two

separate jargons. Applications are expected to determine what an appropriate historical context is and request it either via a bulk transfer or start collecting via a subscribed-to transfer or some combination. In parallel, they should request the current value, and when results from both kinds of operations are in hand, the application can then display, analyze or store them as needed.

Requesting the current value of a Direct Shared Managed Object is simply a matter of invoking the GET method on the attribute in question.

22.5. Control Commands

Control commands will be forwarded by shared managed objects to the responsible peer management system to be executed as though they were local. Attempting to bypass the forwarding process to issue control commands directly to the target object is strongly discouraged for two reasons. First, it may bypass security mechanisms that implement authorization delegation. This may not only prevent access due to missing authorization tokens but may also bypass operations coordination mechanisms resulting in a control tug-of-war and denial of service.

Control results will be published via events and status updates. Direct Shared Managed Objects generally use a simple write such as SNMP or CORBA SET method on the appropriate attribute to implement control, but if a non-SNMP interface is expected, other methods may be defined and used.

22.5.1. Control IDL

The control jargon interface is defined in the following IDL. The included file `nmtypes.idl` is used by those jargons that find it easiest to define their wrapper interfaces in terms of SNMP MIBs. It is defined in section 22.3.4, Common Trend Datatypes.

```
// Declared types

// Each call processes 1 set command which will have a few parameters
// a MIB variable, a MIB value for the variable, and a destination host.

#include "nmtypes.idl"

interface CONTROL {

    exception no_dev_or_oid{};

    // set takes a host ip address, a snmp object identifier
    // (oid), a value to be set for the oid, calls the server and returns
    // the result in output string variable res.

    void set( in DeviceID          dest_host,    // seq of octet
              in MIBVariable      oid,          // seq of ulong
              in MIBValue         val,          // double
              in string            community_name, // community
              out string           res
            );
```

```

// typedefs for downloadFrSvr(); It gets around Corbus limitations
// copied from trend.idl (TFMIBVariable and TFDeviceID).

typedef DeviceID          CJDeviceID;
typedef MIBVariable       CJMIBOID;
typedef sequence<CJDeviceID> seqDevices;
typedef sequence<CJMIBOID>   seqOIDs;

// downloadFrSvr downloads the managed-hosts list and the oids

void downloadFrSvr( out seqOIDs    vars,
                  out seqDevices devices
) raises ( no_dev_or_oid );
};

```

22.6. Event Monitoring

Events will be directed to shared event collections which will forward them to interested subscribing peer management systems.

The Event or Event_PASS API consists of two operations and a distribution or update policy. The operations include a read operation that posts a read request via the PASSREADER library and a write operation, which writes via the PASSWRITER. Distribution policy for individual Event messages is only to write a new Event message if the incoming Event message value is new or different. Event message information is only given to readers if they are newly created (and hence have not seen any of the data yet) or if the reader is already up but there is a newly written Event message. Furthermore, the Event messages may be declared persistent; if the server crashes and restarts, subsequently registered PASSReaders will get all of the current records (per the previous rule).

22.6.1.Event PASSREADER

The Event PASSREADER is a library object that embodies the read distribution policy for Event message information.

22.6.2.Event PASSWRITER

The Event PASSWRITER is also a library object that embodies the write distribution policy for Event message information.

22.6.3.EVENT_PASS IDL

In the standard PASS IDL file pass.idl, and using the standard CORBA IDL mechanism, #include “Event_PASS.idl”, add the following Event refinements as stored in the file

Event_PASS.idl

```
#define PAYLOAD_STRUCT
```

```
struct EVENT_REP {
    string RecName;
    string RecValue;
    string HeartBeat;
};
```

```
};

#define PAYLOAD_REP          EVENT_REP
#define PAYLOAD              EVENT
#define PAYLOAD_PASS        EVENT_PASS
// The following pass-template IDL is described in the section PASS Template IDL
#include "pass-tmpl.idl"
```

22.7. Collateral Service Interfaces

22.7.1. Directory Services

Directory services will be imported from all peer management systems and from the jargon ORB. This will enable management applications to locate Shared Managed Objects and Peer Management System Components.

22.7.2. Time Service

Time services will be imported from the local peer management system. Synchronization with other peer management systems will be automatic if they are all using the same distributed time service and manual (by communications with the remote Point of Contact) otherwise.

22.7.3. Security Services

Authentication and privacy services will be imported from the associated security project. Authorization will be imported from the peer management systems and the jargon ORB.

22.7.4. Management Services

Management Services for the local part of a Shared Managed Object will be imported from the local peer management system. Management services for the CORBA-based jargon components of a Shared Managed Object will be imported from the ORB. Management services for the jargon wrapper components and for coordinated jargon management will be provided. A unified graphic user interface will be used to combine access to and coordinate usage of these management services wherever reasonable. The structure of the management services so provided is defined below in the System Component Description section on Jargon Management.

In addition to providing jargon configuration and GUI monitoring, PASS-based jargons may also define a *heartbeat* attribute in their payloads. The heartbeat is updated by the PASSWriter on a periodic basis and can be used by the PASSReaders to feed status to a *jargon managed object* on the management system importing data via that jargon. If the ultimate source (for example a remote poller) goes down but not the PASSWriter, this can be used to signal the problem across a distribution interface that only transmits changed values such as many PASS configurations do. In addition, PASSReaders may signal problems (such as the PASS object or service going away) directly to a jargon managed object.

To efficiently support collective SMO data such as status or events jargon wrappers may offer various multiplexing services by sharing a single PASS jargon instance across

multiple target shared managed objects. The sharing may be implemented either by using multiple writers to feed the distribution object or by multiplexing inputs and feeding a single writer, or both. Multiple writers are useful if several systems are acting jointly as a single peer. The first system starts up the jargon instance, the others then join in as subsidiary writers. A single writer is more efficient if a single system is serving as the liaison system and can, in its PASSWriter, scan the list of SMOs (either in the platform or internally) and multiplex the necessary inputs. These policies may be stored in environmental attributes, startup command line arguments or in the SMO Share operation implementation.

22.7.5.Persistent Storage Services

Persistent Storage Services will be imported from the source peer management system where appropriate, and from the jargon ORB if appropriate. Persistent Storage includes but is not confined to relational databases, object-oriented databases and flat files. Usage of persistent storage by a jargon may be determined by local policy, in which case appropriate configuration switches should be provided.

22.8. User Interfaces

User interfaces will follow the style of the associated local peer management system as far as possible. When peer system GUI support is missing, X windows compatible application interfaces will be supplied based on Tcl/tk. Tcl/tk scripts, in conjunction with the Tcl/tk interpreter (*wish*) binaries which have been extended to provide access to necessary CORBA and information management system interfaces, will be invoke-able from the information management system's native GUI. The Tcl/tk scripts will be integrated using the information management system's GUI integration tools.

23. System Component Description

The *System Component Description* focuses on a low level detail description of the underlying implementation object model used to support the functionality described in the *Functional Component Description*. All component types and object classes are introduced and discussed and some level of detail is included to describe the critical sub-components of each.

This architecture relies on categorizing structured management information such as MIB data and logs according to their distribution requirements as discussed in the previous sections. In the resulting implementations each jargon consists of one or more code objects, both CORBA objects and undistributed C++ objects. The simple *direct* jargon objects are used for stable, low-volume information, synchronously transferred on demand from the management system. The attribute-snapshot and control commands, which are based on synchronous GET and SET operations to managed devices, are implemented as *direct object attributes*' IDL-generated GETs and SETs.

When information changes often enough or is big enough (or both) that implementing it in a Direct object would be too bandwidth-intensive or prone to hanging due to blocking, it is instead implemented using a *PASS* or *indirect* jargon object, which allows an application to subscribe to a ongoing service. If necessary for efficiency, the jargon may only distribute

the latest changes. PASS objects are defined by a templated IDL that specifies the transport.

There are six jargons for network device management information and functions:

1. The *Status* jargon, a PASS interface to status information, is used for bandwidth efficient monitoring.
2. The *Trend Bulk* jargon interface is used for bandwidth efficient transfer of large stable datasets for providing trend context.
3. The *Trend Subscription*, a PASS interface, is used for bandwidth efficient transfer of large and volatile datasets for providing trend context.
4. The *Snapshot* jargon interface, often is used to get a snapshot of the value of a particular attribute to be combined with the historical context supplied by one of the other Trend jargons.
5. The Control *Command* jargon allows writing directly to the representation of the individual managed objects' stable attributes.
6. The Event jargon is a PASS interface to the traps, events and other asynchronous notifications.

As far as component implementation goes, the platform interface and translation functions (a.k.a. 'wrapper functions') may be a part of either the server operations module (in the case of Direct IDL jargons) or the Reader and Writer modules (in the case of a PASS-based jargon). The wrappers described here are given as examples of the kinds of platform integration issues and potential solutions encountered.

In addition to the jargons, there is a set of jargon management components that serve to route management information and commands to and from jargons. These components consist of 'helper applications' integrated into the local and remote peer information management systems. They are invoked either automatically or manually (depending on the execution architecture) from inside the peer system. Most commonly, the synchronous functions (such as Attribute Snapshots and Control) are invoked by hand from a GUI menu or other command, (or occasionally by a peer system automated poller) and the asynchronous functions (such as Status and Events) are invoked by a combination of peer system notification handlers.

If the peer system itself offers a CORBA-compliant interface, these functions can be incorporated into the shared managed object as a set of standard Object Factory methods (CreateJargon, DestroyJargon, BrowseCurrentJargons) and as ORB service installation functions (RegisterJargonTypes, InstallJargonService). The specifics of the Object Factory and service installation are dependent on the ORB being used.

If there is no CORBA ORB interface or it is not available for a particular platform, an auxiliary ORB, Corbus, is provided and its management API is installed as an external application on the peer information management system. The standard jargon management attributes described in the system interface section above appear as the user-visible application interface.

23.1. Status Transfer

Shared managed objects that wish to export status information need to create a STATUS_PASS Writer, which will in turn locate or, if needed, create a STATUS_PASS instance in the ORB. The STATUS_PASS Writer writes status updates to the STATUS_PASS instance and STATUS_PASS Readers who have subscribed will be notified.

23.1.1.STATUS_PASS

The STATUS_PASS client library and STATUS_PASS server skeleton are generated by the STATUS_PASS IDL. The server operations skeleton file then is modified to implement the STATUS_PASS server functions. Additional payload specific files define status payload utility functions.

23.1.2.STATUS_PASS Writer

When a status snapshot is available (either synchronously, thanks to a local polling mechanism, or asynchronously because of a local status notification mechanism) it is written via a S_P Writer function. If the value is different from the previous one, it is passed on through the STATUS_PASS instance to the STATUS_PASS Reader.

If the Status values must be translated or mapped to a standard value, translator libraries may be linked in. They are invoked automatically via the Translate function. Developers may specify a Null or *NO-OP* library that does no translation or reformatting.

23.1.3.STATUS_PASS Reader

STATUS_PASS Reader clients use a library to subscribe to Status information updates.

If the Status values must be translated or mapped to a standard value, translator libraries may be linked in. They are invoked automatically via the Translate function. Developers may specify a Null or *NO-OP* library that does no translation or reformatting.

23.1.4.TME Unwrapper

Status input to TME is defined by the Status Jargon Management interface (see SMO) on the ManagedNode or Collection via the Import switch, which when Share is invoked starts the reader client. Status is displayed by association with particular resource icons and colors which are invoked by the TME *wputstate* command. No other display is needed because TME relies heavily on the GUI symbol to implement both managed objects and collections of managed objects.

23.1.5.HPOV Wrapper

The SMOViewer Export switch uses *HPLateStart* to invoke the writer client. A first scan of the HPOV database (which is how HPOV implements collections) collects SMOs from the set of all MOs and creates the SMOList collection. Subsequent scans use the SMOList to collect status data from SMOs and feed them to the writer client.

23.1.6.HPOV Unwrapper

Status input to HP OpenView is defined by Status Jargon Management on a Managed Object via the Import switch, which starts the reader client when Share is invoked. Status

is displayed by association with particular resource icons and colors, because HPOV, like TME, relies heavily on the GUI symbol to implement individual managed objects.

23.1.7.TME Wrapper

The SMOViewer Export switch starts (if needed) the writer client when Share is invoked and adds the shared managed object to the writer client process' SMOList.

23.2. *Trend / Historical Context Bulk Transfer*

The Trend Bulk Jargon is a Direct IDL jargon that synchronously transfers Historical Context data. To transfer trend information, clients direct the Trend server to create TrendTable objects, which can then be queried for the desired trend information.

The synchronization of data between a TrendTable and the actual data available from the management platform can be accomplished in several ways. At one extreme, the TrendTable may be populated upon creation and never updated. At the other extreme, the TrendTable may never actually be populated; every request for data goes all the way back to the management system. Between these extremes, the TrendTable may be demand-populated and refreshed whenever the management platform has additional data, like a traditional cache. The choice of strategy depends on a variety of factors such as storage requirements and the interfaces available for extracting trend information from the management platform.

23.2.1.TME Unwrapper

The Policy Region menu bar is extended with the addition of the P2P menu. The Trend item in this menu launches a GUI which allows the user to select trend information to import from a remote management platform. When the trend information is received, the information is written to a file in the TME Brief (one-line) format, and a disabled monitor is added which points to the file.

23.2.2.HPOV Wrapper

The trend server functions as the wrapper for both HPOV and TME. The server determines at run time which platform it is running with. If it is HPOV, the server scans the directory given by \$OV_DB/snmpCollect, where \$OV_DB is an environment variable defined by OpenView, and makes all of the trend information found there available to clients. HP OpenView's Trend information collector, snmpCollect, is manually started as an independent application, and populates the \$OV_DB/snmpCollect directory.

23.2.3.HPOV Unwrapper

The OpenView menu bar is extended with the addition of the P2P menu. The Trend item in this menu launches a GUI which allows the user to select trend information to import from a remote management platform. When the trend information is received, the information is written to a file in a subdirectory under \$OV_DB/snmpCollect (see above).

23.2.4.TME Wrapper

The trend server functions as the wrapper for both HPOV and TME. The server determines at run time which platform it is running with. If it is TME, the server executes a series of TME commands (wlsmon and wlookup) to locate all of the SNMP User-

Defined Numeric monitors, and makes all of the trend information found there available to clients. The user must set up SNMP monitors separately with Tivoli's Sentry/Distributed Monitoring product.

23.3. Trend / Historical Context Subscription Transfer

The Trend Subscription service is also known as the TrendPass service, from its component names.

23.3.1.TREND_PASS

NOTE: The TREND PASS service has not yet been implemented.

The TREND_PASS client library and TREND_PASS server skeleton are generated by the TREND_PASS IDL. The server operations skeleton file is then modified to fully implement the server functions. Additional payload specific files define trend payload utility functions.

23.3.2.TREND_PASS Writer

When a TrendTable entry snapshot is available, it is written via a TREND_PASS Writer function. If the value is different from the previous one, it is passed on through the TREND_PASS instance to the TREND_PASS Reader.

If the TrendTable values must be translated or mapped to a standard value, translator libraries may be linked in. They are invoked automatically via the Translate function. Developers may specify a Null or *NO-OP* library, which does no translation or reformatting

23.3.3.TREND_PASS Reader

TREND_PASS Reader clients use a library to subscribe to TrendTable entry information updates.

If the TrendTable entry values must be translated or mapped to a standard value, translator libraries may be linked in. They are invoked automatically via the Translate function. Developers may specify a Null or *NO-OP* library, which does no translation or reformatting.

23.3.4.TME Unwrapper

TrendSubscription input to TME is defined by the TrendSubscription Jargon Management (see SMOViewer) as either ManagedNode or Collection extension attribute. TrendTables are populated when a Share invoke sees the Import switch and starts the process of subscribing, creates a local TrendTable and starts the TREND_PASSREADER . TrendTables are made visible via a TME Collection Menu hook plus tk windows application TrendViewer.

23.3.5.HPOV Wrapper

HP OpenView's Trend information collector, snmpCollect, is manually started as an independent application. It is manually configured to collect from SMOs by the SMOViewer Export switch and the wrapper tracks the collection by monitoring collection files when Share is invoked.

23.3.6.HPOV Unwrapper

TrendSubscription input to HP OpenView is defined by the TrendBulk Jargon Management on a managed object via the Import command, which starts the reader client via *HPLateStart*. TrendTables are made visible via a toplevel OV installed tk windows TrendViewer application.

23.3.7.TME Wrapper

The Trend Jargon Management's Export command starts an external application formatting of collection information previously configured (by the TME Collection configuration API).

23.4. Basic Shared Managed Object Attribute Snapshot

Attribute Snapshots use the CORBA standard IDL primitive *attribute* to define a synchronous GET function for each attribute whose value is to be collected for a shared managed object.

23.4.1.TME Unwrapper

The Snapshot Jargon Management (see SMOViewer) defines attribute snapshot input to TME as a Collection extension attribute. Attribute display structures are populated by the Import switch, which starts the remote collection process when Share is invoked. Attributes are made visible via a TME Collection Menu hook, plus a tk windows application MIB Browser known as SnapshotViewer.

23.4.2.HPOV Wrapper

The SMOViewer Export switch uses the HP OpenView application startup function *HPLateStart* to invoke the writer client when Share is invoked. A first scan of the HPOV database collects SMOs from the set of all MOs and creates the SMOList. Subsequent scans use the SMOList to collect attribute data from SMOs and feed them to the writer client.

23.4.3.HPOV Unwrapper

The Snapshot Jargon Management defines input to HP OpenView. Snapshot viewer is an external installed tk windows application which reads SMOList and collects MIB data from the Snapshot reader client for each SMO (does not update OV database for that SMO).

23.4.4.TME Wrapper

The SMOViewer Export switch starts (if needed) and adds the shared managed object to the writer client process when Share is invoked. The writer client scans all the SMOs in the SMOList.

23.5. Basic Shared Managed Object Attribute Control

Attribute Snapshots use the CORBA standard IDL primitive *attribute* with the option *read-write* to define a synchronous SET function for each attribute whose value is to be controlled on a shared managed object.

23.5.1.TME Unwrapper

The Control Jargon Management (see SMOViewer) defines control interfaces as a ManagedNode extension attribute. Current values are first populated by a legacy callback on the appropriate Sentry Collection; the Control operation result is used to update the display. The display itself is implemented via a ManagedNode hook plus a tk windows application which is a variant of the SnapshotViewer.

23.5.2.HPOV Wrapper

The SMOViewer Export switch uses *HPLateStart* to start the server when Share is invoked. A first scan of the HPOV database collects SMOs from the set of all MOs and creates the SMOList. Subsequent scans use the SMOList to collect attribute data from SMOs for initial value setting. SET requests are handled as they arrive.

23.5.3.HPOV Unwrapper

The Control Jargon Management defines input to HP OpenView. The Control viewer is a variant of the Snapshot viewer with writing (SETs) enabled.

23.5.4.TME Wrapper

The SMOViewer Export switch starts (if needed) and adds the shared managed objects to the writer client process when Share is invoked. The writer client scans all the SMOs in the SMOList with Control enabled.

23.6. Event Forwarding

NOTE: Event Forwarding has not yet been implemented.

Shared managed objects that wish to export status information need to create a STATUS_PASS Writer, which will in turn create a STATUS_PASS instance. Status updates are written to the STATUS_PASS instance and STATUS_PASS Readers who have subscribed will be notified.

23.6.1.EVENT_PASS

The EVENT_PASS client library and EVENT_PASS server skeleton file are generated by the EVENT_PASS IDL. The server operations skeleton file is then modified to fully implement the server functions.

23.6.2.EVENT_PASS Writer

When an event is available, it is written via an EVENT_PASS Writer function.

In the case of events, the value is always different from the previous one, so it is passed on through the EVENT_PASS instance to the EVENT_PASS Reader.

If the events must be translated or mapped to a standard format, translator libraries may be linked in. They are invoked automatically via the Translate function. Developers may specify a Null or *NO-OP* library, which does no translation or reformatting.

23.6.3.EVENT_PASS Reader

EVENT_PASS Reader clients use a library to subscribe to Events.

If the events must be translated or mapped to a standard format, translator libraries may be linked in. They are invoked automatically via the Translate function. Developers may specify a Null or *NO-OP* library, which does no translation or reformatting.

23.6.4.TME Unwrapper

Input to TME is defined in Enterprise Console Event handling or in SMOViewer as a ManagedNode extension attribute EventLog which is populated by the Import switch starting, when Share is invoked, a legacy callback logfile writer and an EVENT_PASS reader client. The event logs are visible via a ManagedNode menu hook plus a tk windows application EventLogViewer or by forwarding to EnterpriseConsole.

23.6.5.HPOV Wrapper

The SMOViewer Export switch, when Share is invoked, uses *HPLateStart* to invoke the writer client. A first scan of the HPOV database collects SMOs from the set of all MOs and creates the SMOList. Subsequent scans use the SMOList to setup event forwarding from SMOs and feed them to the write.

23.6.6.HPOV Unwrapper

Input to HP OpenView is defined by the Event Jargon Management, which uses the Import switch and the Share function to start the Event reader client. The EventViewer is an external installed tk windows application which accepts event data from members of the SMOList

23.6.7.TME Wrapper

The SMOViewer Export switch and the Share function start (if needed) and add the M.O. to the writer client process which covers all the SMOs in the SMOList with event forwarding enabled.

23.7. Jargon Management

The normally visible Jargon management interfaces consist of managing the the Shared Managed Objects, the local Managed Objects and Collections on which they are based, the local representations of the remote Peer Managers which have custody, and the connecting Jargons. For the most part, the operators deal with the Share/Unshare functions, which create/destroy a SMO. All the other operations are subsidiary.

Access to the SMO and Peer Manager interfaces is provided by P2P Management browsers and control GUIs which can either be a top-level management system menu item (for shared managed objects which are managed as collections) or can be attached to individual SMOs when they require individual jargons.

‘Under the hood’ Jargon management interfaces implement a number of operations policies such as updates-only distribution, writer multiplexing level, persistent storage usage, and security level. In addition they coordinate jargon component startup and interactions with the parent platform components such as pollers and displays.

Access to low-level Jargon management interfaces is via a combination of environment variables, command line arguments and a monitoring and control GUI that breaks out the individual jargon components.

24. Appendices

24.1. On-line References

- [Peer to Peer System Requirements](#) -reqs.html
- [Network Management Study](#) – nmfinal4.html
- [Introduction to Cronus](#) - http://morpheus.bbn.com/cronus_man/
- [Cronus Operator Manual](#) - http://morpheus.bbn.com/cronus_man/operman/cat1/
- [Cronus User Manual](#) - http://morpheus.bbn.com/cronus_man/userman/cat1/
- [Corbus Operator's and Installation Manual](#) - <http://www.bbn.com/offerings/corbus/corbusim.htm/>
- [CORBA 2.0 Specification](#) - <http://www.omg.org/corba/corbiiop.htm>
- [standard SNMP MIB definitions](#) -atommib.html
- [standard SNMP MIB definitions](#) - 1id-abstracts.html
- [Odyssey Research Associates](#) - <http://www.oracorp.com/>
- [SNMPIDL](#) - <http://www.smile.fr/us/prod.htm>
- [XBind](#) - <http://comet.ctr.columbia.edu/xbind/>

24.2. Glossary

24.2.1. Peer to Peer

A communications model that allows but does not require the management systems to define themselves as subordinates or superiors in order to exchange information about the managed objects in their domain.

24.2.2. Jargon

A specialized CORBA-based communications service (and associated API) that translates and distributes management information about shared managed objects to heterogeneous peer management systems using infrastructure objects.

24.2.3. Managed Object

A device or component whose management interface is defined via an object-oriented language such as the SNMP MIB definitions. Also known as a *Managed Object Class* or *MOC*.

24.2.4. Managed Object Instance or MOI

A specific instance of a managed object class, identified by a fully distinguished name (system or node name plus any other labels that will enable it to be distinguished from others of its class.

24.2.5.Shared Managed Object

A distributed managed object, responsibility for which is shared among peer management systems. It is defined as a collection of jargons that form an association from a local managed object to the custodial set of remote peer management systems.

24.2.6.Infrastructure Object

A jargon component which uses object-oriented technology, often CORBA-based, to export or import shared managed object information. There are two main classes of infrastructure objects: distributed objects, which are CORBA-based, and local objects, which are not distributed and use traditional object oriented languages and techniques. Of the distributed infrastructure objects there are two types: direct or synchronous request/response objects and indirect or piecewise asynchronous service (PASS) objects.

24.2.7.Direct Object

A collection of management information and operations from a shared managed object, the API to which is *directly* defined by CORBA IDL. Since CORBA IDL operations are based on synchronous RPC, this family of jargons is useful only for occasional information transfers. Compare with Indirect or PASS Object.

24.2.8.Indirect Object

see PASS Object

24.2.9.PASS Object

A PASS Object, together with the associated reader and writer wrappers, allows a peer management system to subscribe to an update service which only distributes the latest changes in a shared managed object. PASS objects are defined by a templated IDL, which specifies the transport and the standard transfer datatype.

24.2.10.Collateral Services

A set of services which can be defined modularly by their own architectures but whose functions provide necessary utilities for management systems.

24.2.11.Authentication

A security function where the identities of the source and occasionally the destination(s) in an information transfer are checked, usually by cryptographic methods, to ensure they are the source and destination(s) intended.

24.2.12.Authorization

A security function where the right or authority of the requestor to request an operation is checked, usually by comparing the requestor's identity to an access control set or by checking a permissions token included with the request for validity.

24.2.13.Integrity

A security function where the request, response or notification message is checked, usually cryptographically, to detect tampering.

24.2.14.Privacy

A security function where isolation or encryption prevent interception or unauthorized reading of sensitive request, response or notification messages.

24.2.15.Replay Attack

An attempt to fool a destination into accepting as genuine a request, response or notification message that is actually a stale or misleading copy of an earlier, intercepted message.

24.2.16.Denial of Service Attack

An attempt to prevent useful work or communication by others by overloading the capacity of some resource. Distinguished from simple overload by the hostile intent of the source.

24.2.17.Persistent Storage

Keeping information on disk or in another storage medium, which will survive the crash or power failure of the peer to peer system.

24.2.18.Management of Management

Also known by its acronym, MOM, the recursive application of standardized management methods to the management systems and infrastructure. As bootstrapping and graceful shutdown require that the management systems work before most network, system and collateral services infrastructure is up and when they are down, this results in a certain duplication and occasional embedding of some services and increased overhead.

24.2.19.Heartbeat

A management of management function in which communications infrastructure which is only occasionally used or whose traffic is sporadic is periodically checked to see if it is up. The period

is determined by trading off how quickly the information transferred can go stale against the overhead cost of using the channel for MOM information.

24.2.20.XBind

The ATM interface is XBind, a CORBA based management API based on the ATM signaling interface and used to control end to end ATM resources.